THEORETICAL AND SOFTWARE CONSIDERATIONS FOR GENERAL DYNAMIC ANALYSIS USING MULTILEVEL SUBSTRUCTURED MODELS

By

Richard J. Schmidt

Robert H. Dodds Jr.

A Report on Research Sponsored by the
NASA LEWIS RESEARCH CENTER
NASA NAG 3-32

Structural Engineering and Engineering Materials

SM Report No. 15

September 1985

THE UNIVERSITY OF KANSAS CENTER FOR RESEARCH, INC.
2291 Irving Hill Drive—Campus West Lawrence, Kansas 66045

THEORETICAL AND SOFTWARE CONSIDERATIONS FOR GENERAL DYNAMIC ANALYSIS USING MULTILEVEL SUBSTRUCTURED MODELS

by

Richard J. Schmidt

and

Robert H. Dodds., Jr.

A Report on Research Sponsored By

NASA Lewis Research Center

Research Grant NAG3-32

University of Kansas

Lawrence, Kansas

September 1985

50272-101

REPORT DOCUMENTATION PAGE	1. REPORT NO.	2.	3. Recipient's Accession No.

4. Title and Subtitle

Theoretical and Software Considerations for General Dynamic Analysis Using Multilevel Substructured Models

5. Report Date

September 1985

6.

7. Author(s)

Richard J. Schmidt and Robert H. Dodds, Jr.

8. Performing Organization Rept. No.

SM Report No. 15

9. Performing Organization Name and Address

University of Kansas Center for Research Inc.
2291 Irving Hill Drive, West Campus
Lawrence, Kansas 66045

10. Project/Task/Work Unit No.

11. Contract(C) or Grant(G) No.

(C)

(G)

12. Sponsoring Organization Name and Address

13. Type of Report & Period Covered

final

14.

15. Supplementary Notes

16. Abstract (Limit: 200 words)

An approach is presented for the dynamic analysis of complex structural systems using the finite element method and multilevel substructured models. The fixed-interface method is selected for substructure reduction because of its efficiency, accuracy, and adaptability to restart and reanalysis. This method is extended to reduction of substructures which are themselves composed of reduced substructures. Emphasis is placed on the implementation and performance of the method in a general-purpose software system. Solution algorithms consistent with the chosen data structures are presented in detail.

This study demonstrates that successful finite element software requires the use of software executives to supplement the algorithmic language. As modeling and analysis techniques become more complex, proportionally more implementation effort is spent on data and computer resource management. Executive systems are essential tools for these tasks. The complexity of the implementation of restart and reanalysis porcedures also illustrate the need for executive systems to support the non-computational aspects of the software.

The example problems show that significant computational efficiencies can be achieved through proper use of substructuring and reduction techniques without sacrificing solution accuracy. The unique restart and reanalysis capabilities developed in this study and the flexible procedures for multilevel substructured modeling allow analysts to achieve economical yet accurate analyses of complex structural systems.

17. Document Analysis a. Descriptors

dynamic analysis, finite element method, multilevel substructuring, modal synthesis, eigenproblem solution, subspace iteration, software executives, software engineering

b. Identifiers/Open-Ended Terms

c. COSATI Field/Group

18. Availability Statement	19. Security Class (This Report)	21. No. of Pages
release unlimited	unclassified	188
	20. Security Class (This Page) unclassified	22. Price

(See ANSI-Z39.18) *See Instructions on Reverse*

OPTIONAL FORM 272 (4-77)
(Formerly NTIS-35)
Department of Commerce

ACKNOWLEDGEMENTS

This report is based on the dissertation of Richard J. Schmidt submitted to the Department of Civil Engineering, University of Kansas for the degree of Doctor of Philosophy. The study was conducted under the direction of Dr. Robert H. Dodds, Jr.

The research was supported by the NASA Lewis Research Center under Grant No. NAG3-32. Dr. Murray S. Hirschbein served as NASA Technical Officer.

Numerical computations were performed on Harris computers operated by the Computer Aided Engineering Laboratory, School of Engineering, University of Kansas.

CONTENTS

CHAPTER Page

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1 -- INTRODUCTION

1.1 General

The finite element method (FEM) provides the most general modeling procedure for dynamic analysis of complex structural systems. Often it is necessary to obtain detailed displacement, strain, and stress information within a small region or over an entire structural model. Such a requirement is usually satisfied by dividing the model into a large number of elements. As the number of elements increases, the cost of the analysis correspondingly increases. Limitations of computing resources (both hardware and software) may force the analyst to compromise his objectives by restricting the degree to which the model is refined. Substructuring is a modeling technique which relieves many of the restrictions on model refinement.

Substructuring with static condensation [15, 20, 50] is a popular technique for improving efficiency of static analysis. In the substructuring technique, unique or functionally distinct portions of a structural system are analyzed separately, condensed, and then combined to form a reduced model. This reduced model, having fewer degrees of freedom (DOF), is generally more economical to analyze than the original structural model. Results from a static analysis of a condensed model are identical to those for the same model without condensation. Static condensation is therefore termed an "exact" reduction procedure.

Aside from its computational advantages, the substructuring technique yields a secondary benefit. Independent development of the various structural components (substructures) can proceed

simultaneously. The structural frame of an aircraft provides the classic example. Independent design teams develop the individual substructures: wing assembly, fuselage sections, vertical stabilizer, etc. The substructures are later interfaced at their common boundaries. The modeling and analysis technique can be extended using multilevel substructuring, where the individual substructures can themselves be composed of condensed substructures.

In dynamic analysis, exact reduction of an individual substructure is dependent upon the natural frequencies of the total structural system. Since the system frequencies are objectives of the analysis and as yet unknown, the analyst must use reduction methods that are either iterative or frequency independent (and therefore approximate). The various reduction methods are collectively known as procedures for component mode synthesis or modal synthesis.

In general, modal synthesis techniques have not been incorporated into general FEM programs [13]. A possible exception is some work on proprietary computer codes, full details of which are not readily available. Analyses presented in the literature based on modal synthesis techniques have been achieved by combining the functions of structural modeling, eigensolution, and matrix manipulation through the use of a number of independent and highly specialized computer programs. As a consequence of this lack of sophistication in available software, only trival models have been studied (e.g., planar trusses, rectangular plates, etc). Each analysis requires a specialized driver program to manage the computational procedures unique to the individual structural model. Clearly, a more general analysis procedure is required to permit general studies of modal synthesis techniques.

The objectives of this work are to review the state-of-the-art in modal synthesis; to design and implement a general, user-oriented software system incorporating multilevel substructured modeling for dynamic analysis; and to perform preliminary evaluations of the impact of the modeling and analysis techniques on computed results. The development of general-purpose analysis systems, using sophisticated software techniques, is vital to the incorporation of new analytical techniques into the analysis and design procedures used by practicing engineers and researchers. Modal synthesis techniques must be included as an integral part of the dynamic analysis capabilities of general FEM software. Without general-purpose analysis systems, the burden of developing an individual analysis program for each unique structure would significantly outweigh the computational advantages available with modal synthesis.

1.2 Substructured Modeling Techniques

A brief review of the substructuring and condensation prodecures for static analysis is needed before modal synthesis techniques can be reviewed. Many investigators [20, 50, 56] have shown that partitioning of a structural model into smaller, often identical, substructures can lead to significant savings in model generation and computer solution costs for static, linear and nonlinear analysis. The choice of partitions is generally guided by economic, fabrication, or symmetry constraints. The boundaries which result between substructures due to partitioning may then be either real or artificial in form. When the structure partitioning is applied to an assembly of substructures, a recursive procedure known as multilevel substructuring is established.

The substructure partitioning ends when all "lowest level" structures are composed of only finite elements.

The organization of the structural hierarchy for a multilevel substructured model is represented as an inverted tree. The top of the tree (the root node) defines the highest level structure and resides at level "n" of the hierarchy. Any number of substructure levels may be defined below the root node. There is no theoretical limit on the number of branches (or elements) that enter a node (or structure) at level "i" from level "i-1". All terminal nodes of the tree are finite elements (ex: bars, frames, triangles, etc.). For generality, no distinction is made throughout the hierarchy between finite elements and substructures.

For static analysis of both linear and nonlinear structures, it has been shown that a substructured model yields the same solution as a nonsubstructured model which contains only finite elements. The equations governing the substructuring technique are fully documented elsewhere [53] and will not be reviewed here. Instead a small example is presented which illustrates the terminology associated with the substructuring technique and the degree of simplification possible with a user-oriented approach to substructure analysis. The example structure is a simple two-span, plane truss shown in Figure 1.1. Components of the substructured model are shown in Figure 1.2, with names assigned to each component for identification purposes. Figure 1.3 illustrates the substructure hierarchy in inverted-tree form. The lowest-level structure is the hierarchy is SPAN.

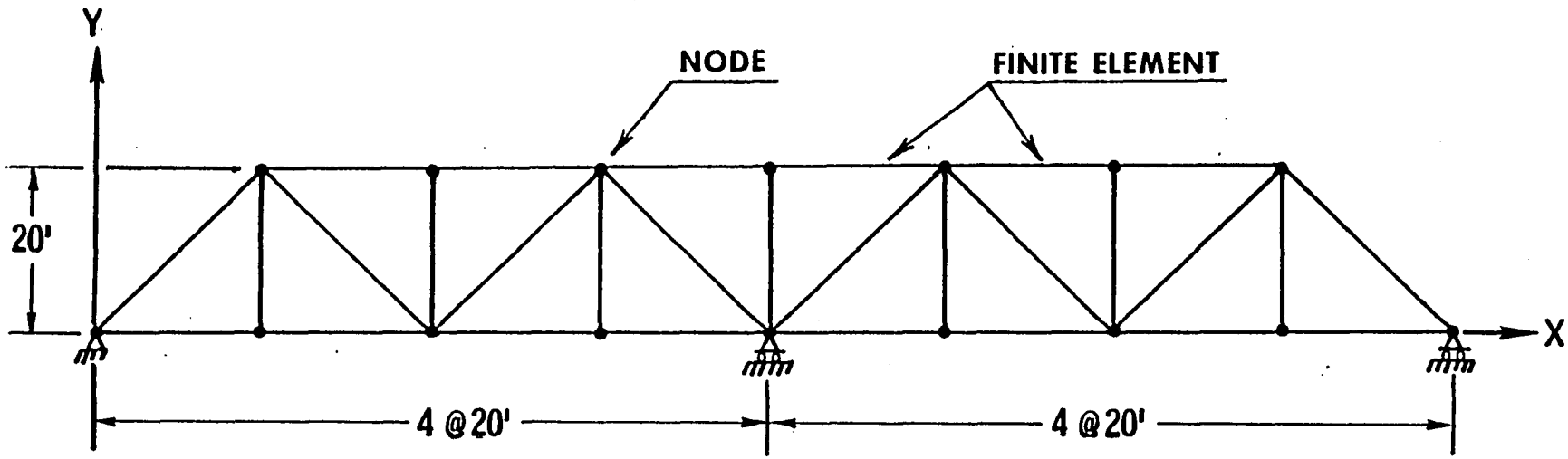


Figure 1.1. Bridge Structure to Illustrate Substructuring

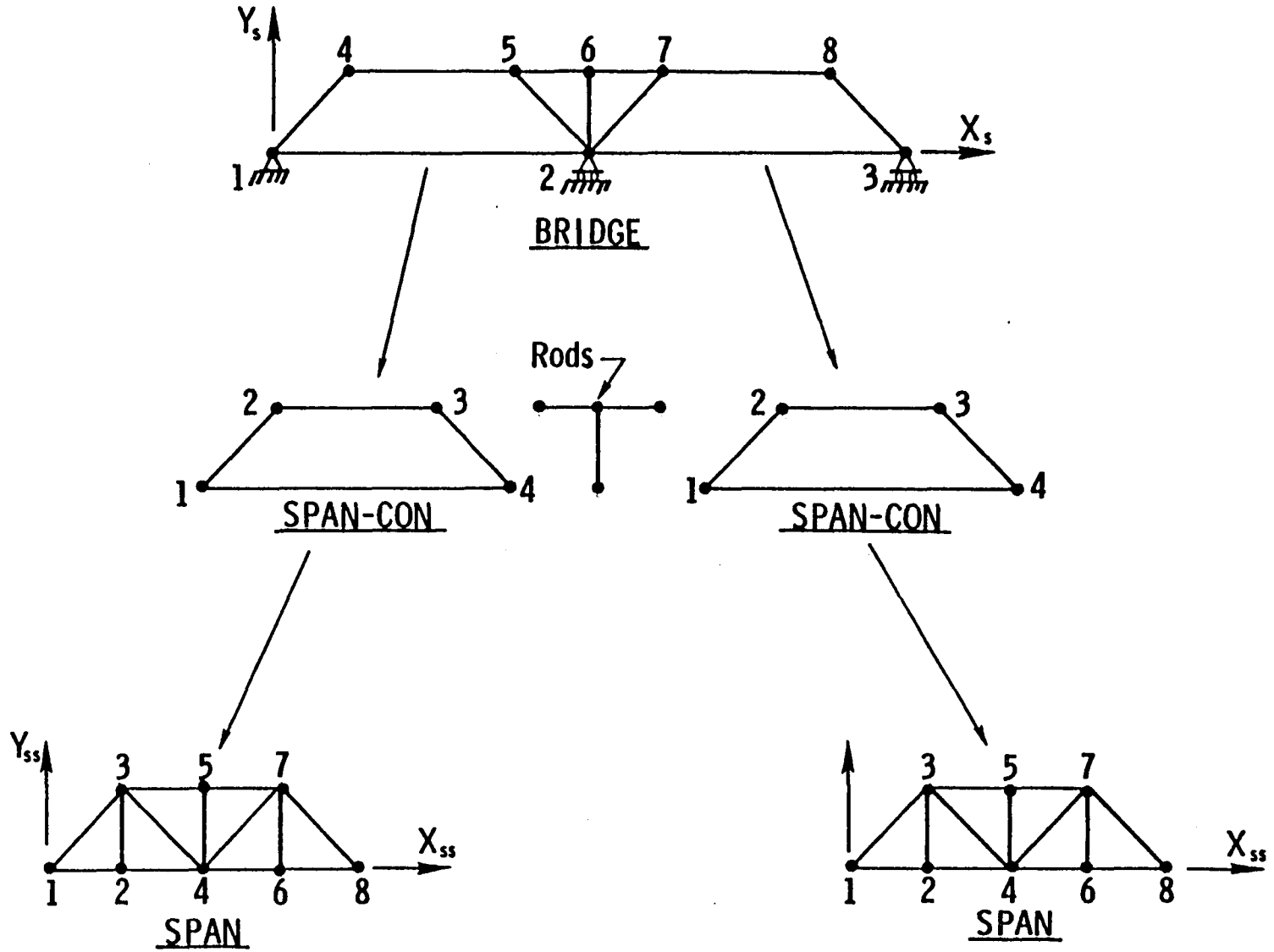


Figure 1.2. Substructured BRIDGE Model

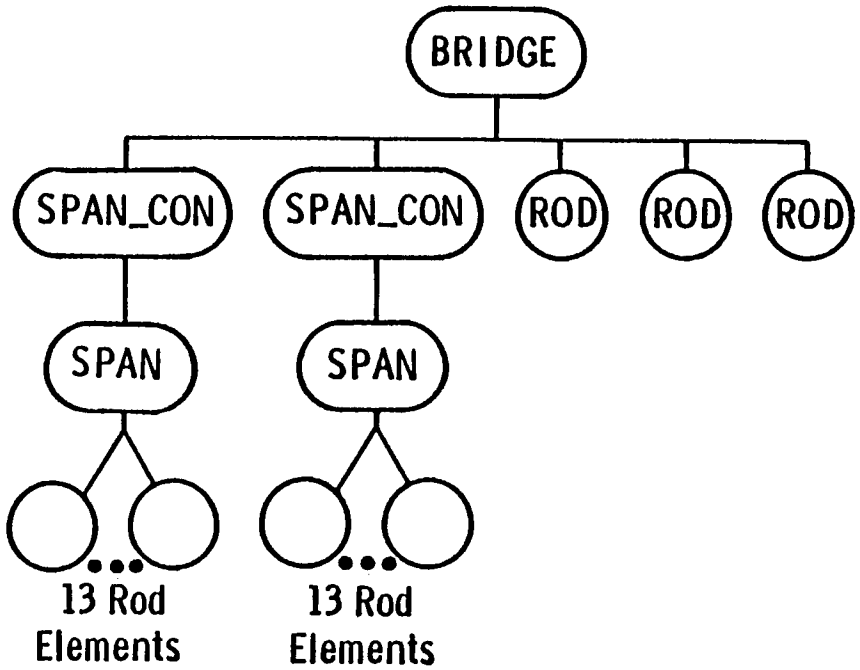


Figure 1.3. Structural Hierarchy for BRIDGE Model

After substructure SPAN is defined, nodes 1, 3, 7, and 8 are selected as boundary nodes. They are retained in the condensed substructure SPAN_CON for connection to adjacent substructures. The remaining interior nodes (2, 4, 5, and 6) are eliminated by condensation. The transformation of coordinates from SPAN to SPAN_CON is achieved by use of the static constraint modes. A static constraint mode is the displaced configuration of the interior nodes when a unit displacement is applied to one of the boundary nodes with all other boundary nodes constrained. The highest level structure, BRIDGE, is defined using two copies of the condensed substructure and three rod elements for closure over the center support.

Figure 1.4 illustrates the ease with which this structural model is defined for analysis. The problem oriented language (POL) used to describe the model is taken from the POLO-FINITE structural mechanics system. As described below, POLO-FINITE supports user-defined, multi-level substructuring as a natural extension of standard modeling and analysis procedures. The lowest level substructure, SPAN, contains 8 nodes and 13 elements. Element types, properties, topology, and nodal coordinates are easily defined through the POL. The condensed version of SPAN is then defined as structure SPAN_CON. Structure SPAN_CON contains the four boundary nodes from structure SPAN. These nodes are identified through the incidence list for SPAN_CON. Structure SPAN is referred to as the "parent" structure. SPAN_CON is the "child". This technique for defining the condensed structure at an intermediate level in the hierarchy eliminates confusion on the analyst's part and maintains a consistent definition of structures throughout the hierarchy.

Structure BRIDGE is modeled from two copies of SPAN_CON and three additional rod elements. Copies of SPAN_CON (elements 1 and 2) are

```
*RUN FINITE
C
C          DEFINE THE BRIDGE SEGMENT:  SPAN.
C          UNITS ARE "KIPS" AND "FEET".
C
  STRUCTURE SPAN
    NUMBER OF NODES 8 ELEMENTS 13
    ELEMENTS ALL TYPE ROD  E 3.0E04  AX 0.0347
    COORDINATES
      1     0.0   0.0
      2    20.0   0.0
      3    20.0  20.0
      4    40.0   0.0
      5    40.0  20.0
      6    60.0   0.0
      7    60.0  20.0
      8    80.0   0.0
    INCIDENCES
      1   1   3
      2   2   3
      3   3   4
      4   4   5
      5   4   7
      6   6   7
      7   7   8
      8   3   5
      9   5   7
     10   1   2
     11   2   4
     12   4   6
     13   6   8
C
C              END OF STRUCTURE SPAN
C
C
C              DEFINE THE CONDENSED VERSION OF STRUCTURE SPAN.
C              RETAIN NODES 1 3 7 AND 8 IN THE CONDENSED STRUCTURE.
C
  STRUCTURE SPAN_CON
    NUMBER OF NODES 4 ELEMENTS 1
    ELEMENT 1 TYPE SPAN CONDENSED
    INCIDENCES
      1   1  3  7  8
C
C              END OF STRUCTURE SPAN_CON
C
```

```
C
C              DEFINE THE HIGHEST LEVEL STRUCTURE AS A COMBINATION
C              OF TWO CONDENSED SPANS AND THREE SIMPLE ROD ELEMENTS
C
C
  STRUCTURE BRIDGE
C
    NUMBER OF NODES 8  ELEMENTS 5
    ELEMENTS
      1 2  TYPE SPAN_CON ROTATION SUPPRESSED
      3-5  TYPE ROD E 3.0E04  AX 0.0347
C
    COORDINATES
      2     0.0    0.0
      5   -20.0   20.0
      6     0.0   20.0
      7    20.0   20.0
C
    INCIDENCES
      1     1  4  5  2
      2     2  7  8  3
      3     5  6
      4     6  7
      5     2  6
C
    CONSTRAINTS
      1-3  V  =  0.0
      1    U  =  0.0
C
    <definition of loads>
C
    <requests for computation>
C
    <requests for output>
C
    STOP
```

Figure 1.4. POL Definition of BRIDGE Model

placed into BRIDGE without rotation from the coordinate system in which they were defined. The three rod elements require nodal coordinates for computation of element size and orientation. Boundary constraints on BRIDGE impose the simple support boundary conditions illustrated in Figure 1.1. Definition of loads and requests for computation and output follow.

1.3 Modal Synthesis Techniques

The selection of a method for reducing the order of a structural model is a key step in the design of a software system with dynamic analysis and substructuring capabilities. In dynamic analysis, reduction is applied to both the stiffness and mass matrices of the structural model. Static condensation is an "exact" method for reducing the size of the stiffness matrix in static analysis. The method is termed "exact" because results for a substructured model are identical to results for a model of the same structure in which substructuring is not used. Chapter 2 will show that no corresponding exact reduction method exists for dynamic analysis. Numerous dynamic reduction techniques have been proposed to improve substructure representations [1, 2, 6, 10-14, 21-34, 36-40, 44-49, 52, 54, 59]. The remainder of this section presents an overview of the most significant techniques available in the open literature in order to illustrate the variety of procedures that have been developed.

The concept of component mode synthesis was first proposed by Hurty in 1960 [31]. Hurty developed a method to aid in the analysis of framed structures. Continuous beam members were represented by admissible functions (low-order polynomials) to develop a numerical model with a

finite number of degrees of freedom. This procedure is essentially the application of the Rayleigh-Ritz procedure at the member level.

Perhaps the simplest of all component mode techniques is Guyan reduction [23]. This approach is a direct extension of static condensation. The transformation matrix of static constraint modes, which is used to reduce the order of the stiffness matrix in static analysis, is also used to reduce the order of the structure mass matrix. The kinetic energy of the interior nodes is represented by only static mode shapes, thus creating the potential for significant analysis error. The simplicity of this method makes it the most popular reduction technique in use today in spite of its limited accuracy.

Improved recovery of the substructure displacements for models condensed by Guyan reduction was studied first by Kidder [36] and again later by Miller [48]. Reduction of the substructure equations follows the standard procedure for Guyan reduction. After the mode shapes for the highest level structure have been computed, a frequency dependent transformation is written between the retained DOF at the highest level and the reduced substructure DOF. This new transformation is then used to recover substructure mode shapes. The frequency estimates obtained by eigenvalue solution of the system equations are used in the new transformation.

Rational procedures for selection of retained (or master) DOF for Guyan reduction were proposed by Henshell and Ong [26] and later by Shah and Raymund [54]. The objective of the procedures is to retain those DOF that most closely maintain the low-frequency response of the structure. The first approach simply retains the DOF with the smallest stiffness-to-mass ratios. The other approach is to recursively perform

Guyan reduction, eliminating only one DOF at a time. The DOF that is eliminated is presumed to have a negligible effect on the frequency range of interest. The use of these methods is limited to models that are not substructured but that are large enough such that reduction prior to eigenproblem solution is desirable.

Hurty's component mode method was extended in 1965 to include discrete (finite element) models [32]. Instead of low-order polynomials, physical mode shapes that represented the individual substructure are used as component modes. These mode shapes are rigid-body modes, static constraint modes, and fixed-fixed normal modes. A fixed-fixed normal mode is a vibration mode shape for an individual substructure with all of its boundary nodes constrained. Rigid-body modes are a substructure's displacement configurations which contain no strain energy. The various mode shapes are illustrated in Figure 1.5. Geometric compatibility between adjacent substructures is enforced through constraint equations at nodes common to both substructures. A simplification of Hurty's fixed-interface method was presented by Craig and Bampton [10]. Substructure component modes were divided into only two groups: constraint modes and normal modes. This resulted in a procedure which is conceptually simpler, easier to implement in the analysis software, and easier for the analyst to use.

Goldman [22] introduced the free-interface method in which only rigid-body modes and free-free normal modes are used in the synthesis process. This technique eliminates the computation of static constraint modes, but this advantage is negated by the poor accuracy of the method. Hou [30] presented a variation of Goldman's free-interface method in which no distinction is made between rigid-body modes and free-free

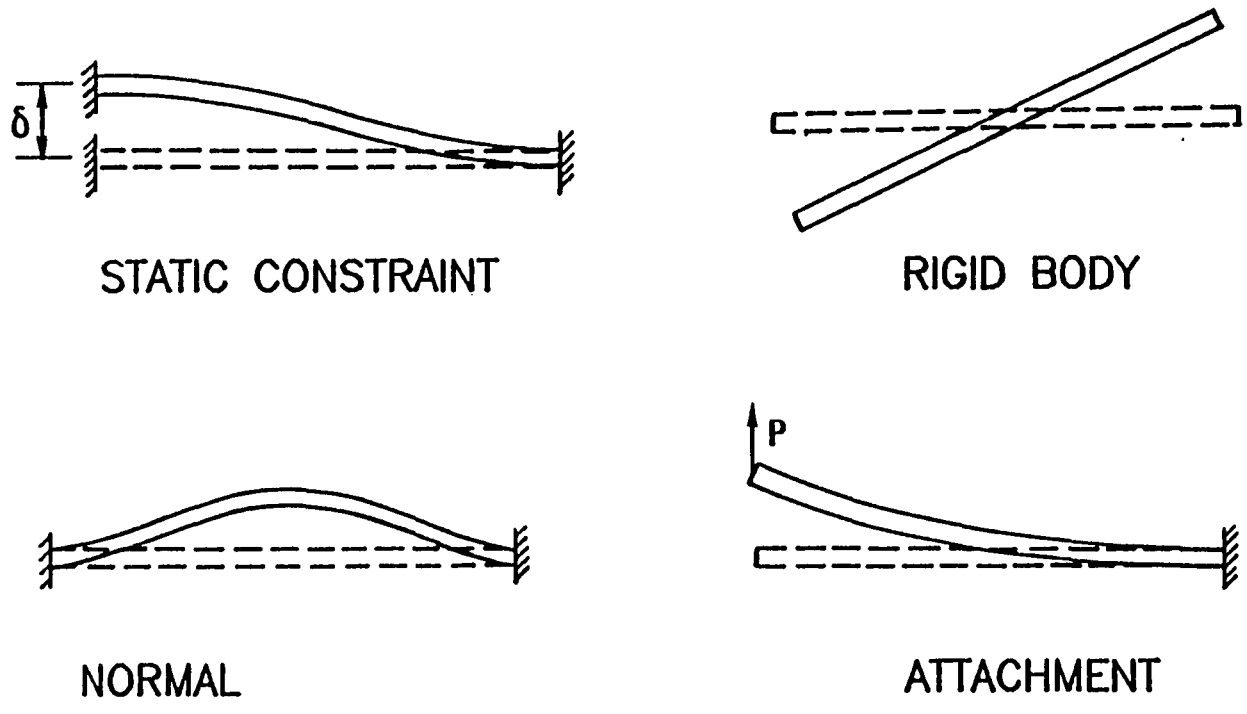


Figure 1.5. Typical Component Modes

normal (elastic) modes. Hou's approach also includes an error analysis procedure to evaluate convergence.

Gladwell [21] introduced "branch mode analysis" by combining free-interface and fixed-interface analyses to reduce the order of the coefficient matrices (stiffness and mass) for individual substructures. The reduction procedure depends upon the topologic arrangement of the substructures in the model. Thus reduction of any one substructure requires knowledge of the arrangement of all substructures in the model.

Bajan, et. al. [1] developed an iterative form of the fixed-interface method. Significant improvements in accuracy are achieved by repeating the reduction, based on updated estimates of system frequencies and mode shapes.

A second-order, frequency-dependent Guyan reduction procedure was developed by Wright and Miles [59] to improve the accuracy of the reduction. Use of this approach requires the solution of a non-symmetric eigenvalue problem which has twice the order of a standard Guyan reduction model.

A modification of the free-interface method known as "interface loading" was presented by Benfield and Hruda [6]. Interface loading is essentially a modification of the stiffness and mass of the object substructure to account for adjacent substructures prior to computation of free-free component modes for the object substructure. Morosow and Abbott [49] developed a mode selection procedure applicable to interface loading. Holze and Boresi [29] incorporated the interface loading technique into a complete dynamic analysis procedure.

An approach for improving displacement recovery for modal synthesis models, similar to that used in Guyan reduction, was presented by Kuhar

and Stahle [38]. They developed a dynamic transformation method which can be used with any of the basic modal synthesis methods. A frequency dependent transformation of the equations for the highest level structure is developed following modal synthesis reduction. The transformation is derived for a given target frequency about which further reduction is desired. Improvements in substructure mode shape recovery are achieved by using the frequency dependent transformation after system modes have been computed.

The use of attachment modes as an additional type of component mode was first proposed by Bamford, et. al. [2]. An attachment mode is the displaced configuration of a substructure when a unit force is applied to one boundary DOF while all other boundary DOF remain free of loads. Use of attachment modes was expected to reduce the number of normal modes required to accurately describe the displacement behavior of the substructure. However, there is a potential problem of linear dependence between attachment modes and normal modes.

MacNeal [44] introduced the use of hybrid modes and inertia-relief attachment modes for component mode synthesis. Hybrid modes are substructure normal modes computed with a combination of fixed and free boundary conditions. Inertia-relief attachment modes are attachment modes for substructures that contain rigid-body freedoms. MacNeal's development also included "residual flexibility" and "residual mass" to approximate the static contribution of the truncated higher modes to the total system response. Use of these residuals is analogous to the mode acceleration procedure for mode superposition. Rubin [52] extended the residual flexibility approach for free-interface substructures by the use of a second-order representation of the truncated modes.

From a more application-oriented viewpoint, Hintz [28] grouped combinations of the four mode classes: rigid-body, static constraint, normal, and attachment into five different interface mode sets. Implications of truncating a selected interface mode set were discussed and guidelines were developed for retaining accuracy with a reduced size model. In another application paper, Craig and Chang [12] discussed alternatives for reduction of boundary coordinates for a number of different modal synthesis methods. Also included in their discussion were requirements for substructure modeling that facilitate experimental verification of the numerical model.

In the only known discussion of modal synthesis for multilevel substructured models, Herting [27] presented work in progress on NASTRAN. The modeling technique allows retention of an arbitrary set of substructure normal modes (fixed, free, or hybrid), inertia relief modes, and all geometric coordinates at substructure boundaries. This method is the most general of the modal synthesis techniques. It is shown in the study that both the fixed-interface method of Craig and Bampton and the MacNeal's residual flexibility method are special cases of the general technique. No discussion of solution economy or user-interface in the NASTRAN implementation are presented.

A pair of frequency-dependent, iterative methods was developed by Leung [39, 40] as extensions of Guyan reduction and the fixed-interface method. In both methods, the unknown system frequency is retained in the substructure reduction equations. Initial estimates for the natural frequencies of interest are improved after each iteration of the procedure. The reduction yields a single coefficient matrix, the dynamic-stiffness matrix, which defines a "standard" eigenvalue problem. In

contrast, other modal synthesis techniques produce two coefficient matrices, generalized stiffness and mass matrices, which define a "generalized" eigenvalue problem.

A second-order substructure condensation procedure generally applicable to the basic modal synthesis methods was presented by Kubomura [37]. In this procedure, the component modes used in reduction include fixed-interface, free-interface, and hybrid modes. Using the system eigenvalue of interest, a rational approach to mode selection is developed.

As an extension of Hurty's first paper on modal synthesis, Meirovitch and Hale expanded the use of admissible functions in component mode synthesis [24, 25, 45-47]. Their work broadened the definition of admissible functions that are suitable for use in substructure reduction. Their technique is applicable to both continuous and discrete structural models. While the use of admissible functions other than eigenfunctions presents the potential for significant reduction in analysis costs, the selection of suitable functions (low-order polynomials) has not been automated such that the approach can be used in a general finite element code.

1.4 Objectives and Scope

The objectives of this work are:

1. To identify those modal synthesis techniques that are suitable for incorporation into a general-purpose FEM software system which includes multilevel substructured modeling capabilities.

2. To design and implement the software required to perform general purpose dynamic analysis. Specific needs include a flexible input language, an automatic and accurate modal synthesis technique, and efficient analysis-restart capabilities.

3. To evaluate the performance of the modal synthesis technique chosen for implementation. Performance is evaluated in terms of the quality of the computed results obtained from models in which condensation is used and the efficiency with which the analyst can restart the analysis to improve the computed results.

While more than one of the foregoing modal synthesis methods may be suitable for multilevel substructured modeling, only the fixed-interface method of Craig and Bampton is studied in detail and implemented. Reasons for this selection and details of the method are presented in Chapter 2. Performance of the condensation technique is evaluated through comparisons of natural frequencies, mode shapes, and strains. The quality of the results for substructured models is measured against results for structural models analyzed without substructuring. Changes in modeling parameters such as substructure topology, degree of reduction, and number of levels in the substructure hierarchy are evaluated for their effect on analysis cost and solution accuracy.

The simple finite elements (trusses, plates, shells, etc.) used in this study are those commonly found in general finite element codes. Elements with shape functions specially suited to dynamic response are not used. However, both lumped and consistent mass formulations for the individual elements are considered.

The computation of element strains is derived directly from vibration mode shapes rather than from the displacement configuration corresponding to a given loading condition. The use of these "modal strains" facilitates a more precise evaluation of the effects of modal synthesis. Errors in strain from each of the vibration modes become recognizable whereas they would be masked if the various mode shapes

were superposed into a single displacement vector. Linear elastic behavior is assumed so element stresses will exhibit the same error characteristics as element strains and need not be evaluated separately.

The effects of structure damping are not included in this study. While synthesis techniques are applicable to reduction of the damping matrix, sufficiently reliable and complete performance studies can be made with damping neglected.

It is recognized that forced-vibration response analysis (mode superposition or time-history integration) is a necessary analytical feature in general purpose dynamic analysis. However, such a capability is not necessary to satisfy the objectives of this study.

Displacement boundary conditions applied to structural systems can take the form of absolute constraints or relative (multi-point) constraints. An absolute constraint imposes a fixed displacement (zero or nonzero) to a specific nodal DOF, regardless of loads on the structure. A relative constraint defines a linear relationship between the displacements of two or more nodal DOF. In this study, only zero-displacement, absolute constraints are considered.

The software developed for this study was implemented as an extension of the POLO-FINITE system [16, 43]. POLO-FINITE is a general finite element system that supports user-defined, multilevel substructured modeling for linear and nonlinear analysis of static systems. The data base management system and the efficient hypermatrix solution techniques make POLO-FINITE a reliable basis on which to develop the above solution capabilities.

A principal feature of the software developed in this study is the analysis restart capability. Since substructuring and condensation for

dynamic analysis is an approximate technique, the analyst will generally desire to verify the model by additional refinement and reanalysis. An efficient software system permits the analyst to simply enhance the existing model and recompute only those quantities affected by the enhancement. This feature is rarely available in an automated, user-controlled form. In this study, analysis restart has no relation to the checkpoint/restart procedures supported by various hardware and software systems.

The remainder of this report is divided into chapters which discuss the major topics covered. Chapter 2 contains a detailed review of the fixed-interface method and its use in multilevel substructured modeling. Details of the POLO executive system as a tool for software development are presented in Chapter 3. Both the development and the run-time environments supported by POLO are reviewed as they pertain to this study. Software design and implementation are discussed in Chapter 4. Topics include the structural modeling procedure, solution algorithms, and analysis restart. The integration of data structures, system processing modules, and element routines are discussed from the viewpoint of the software engineer. Performance of the software resulting from this work is examined in Chapter 5. Results from a number of example problems are discussed. Chapter 6 presents a summary of the study and conclusions. Topics for further investigation are also proposed.

CHAPTER 2 -- FIXED-INTERFACE METHOD

2.1 General

The modal synthesis method selected for implementation in this study is the fixed-interface method as formulated by Craig and Bampton [10]. The reasons for this selection are presented in the next section. Section 2.3 contains a detailed review of the development of the method and the necessary extensions of the method for use with multilevel substructured modeling. Procedures for analysis restart are also developed.

2.2 Features of the Fixed-Interface Method

The goal of the fixed-interface method, as for all of the various modal synthesis methods, is to generate stiffness and mass matrices that accurately represent the stiffness and inertia characteristics of a substructure with the minimum number of degrees of freedom (DOF). Two basic operations are performed in the reduction process. First, the substructure coefficient matrices are transformed from geometric coordinates to a reduced set of generalized coordinates. The transformation matrix normally contains substructure mode shapes that adequately describe the dynamic characteristics of the substructure. The second operation is the assembly of the reduced substructure matrices into the next higher level of the model hierarchy. The details of this operation vary according to the nature of the generalized coordinates representing each substructure. In a multilevel substructured model, the transformation and assembly processes are performed recursively at each level.

In the fixed-interface method, all static constraint modes and some of the fixed-fixed normal modes are selected as component modes for the reduction transformation. The set of generalized coordinates contains normal DOF associated with the fixed-fixed normal modes and boundary DOF which are linked to the static constraint modes. During assembly of the reduced substructures, displacement compatibility is enforced by equations of constraint which tie common boundary DOF at the interfaces between adjacent substructures. Since the boundary DOF retain their physical distinction during the transformation to generalized coordinates, the assembly procedure is identical to that used for non-substructured models. The normal DOF are not included in the constraint equations. A complete development of the method follows in section 2.3.

2.2.1 Efficiency of the Reduction Method

The efficiency of a dynamic reduction method is influenced by three factors. First, the method must produce an accurate reduction in the order (number of DOF) of the substructure stiffness and mass matrices. An efficient method yields synthesized stiffness and mass matrices that accurately represent the dynamic characteristics of the substructure with the minimum number of DOF. Second, the degree of analyst participation should be limited to simply the definition of the model and specification of the solution type. A method should be automatic once the solution process begins, hence eliminating the need for the analyst to interpret intermediate results and restart the analysis. This is not to imply that the analyst should surrender control of the solution process. Instead, the analyst should be relieved of the burdensome task of supervising the computational process. Third, the synthesis method

should be efficient in its use of computer resources. Given the problem size, algorithms should be chosen that minimize the required computer resources, particularly processor time and I/O (data transfers to and from secondary storage). The number of arithmetic operations performed should be predictable rather than dependent upon an arbitrary test for convergence of an iterative process.

The fixed-interface method successfully satisfies the efficiency criterion. The method is simple to apply and yields a significant size reduction of properly substructured models. As will be demonstrated in the example problems, the required user input and control is minimal.

2.2.2 Applicability to General Problems

A wide variety of dynamics problems exists for which modal synthesis is needed to achieve an economical and accurate solution. A synthesis method used in a general purpose FEM system should be capable of modeling substructures over a broad range of geometries with various types of boundary constraint. Also helpful would be the ability to incorporate experimental data (natural frequencies and mode shapes) into the substructured model.

Dynamic reduction methods should lend themselves to incremental solution procedures. By necessity, finite element analysis of a nonlinear structure is performed incrementally. As the effects of nonlinear materials and geometry occur, the coefficient matrices must be reformulated to accurately model the current state of the structure.

The fixed-interface method has limited capability to use experimental data. In the computation of substructure mode shapes for the reduction process, all boundary nodes are fixed. As a consequence,

experimental testing of a structural component must match this boundary constraint if the results are to be useful in the finite element model. The fixed boundary conditions may prove impossible to develop experimentally. However, the fixed-interface method is well suited to the iterative solutions in nonlinear analysis. When linear substructures are reduced and act as elastic restraint to the nonlinear region of a model, the iterations can be performed efficiently with no need to continuously repeat computations for the linear regions.

2.2.3 Substructure Independence

The analysis and design responsibilities of the various components of a structure are often distributed among different organizational groups. This separation of responsibilities has many advantages and should not be encumbered by the synthesis method used in analysis. Substructure independence also provides computational advantages. The definition and analysis of individual substructures establishes natural breakpoints in the analysis process, allowing the analyst to "step through" a complex model, one component at a time. Also, use of efficient parallel processing hardware is possible when individual substructures are treated independently. Therefore, the synthesis method should treat each unique substructure as an isolated entity in evaluating its dynamic response prior to system assembly. The topology that defines substructure connectivity should not be required until the equations at the next level of the hierarchy are ready for assembly.

Substructure independence is preserved in the fixed-interface method. The requirement for fixed boundary nodes, that is a drawback with respect to the previous criteron, is the key factor in satisfying

substructure independence. Consideration of substructure topology is not necessary in the condensation and synthesis of a given substructure.

2.2.4 Ease of Reanalysis

The most reliable test for convergence of a dynamic reduction method requires a second solution of the problem with a more highly refined model (more independent DOF). The addition of more DOF to the model can be a relatively simple task, achieved at little expense, or it can be as difficult and expensive as a complete reanalysis of each substructure. The ideal synthesis method allows the simple addition of previously neglected terms to improve the accuracy of the reduction. These terms generally take the form of truncated substructure normal modes.

As demonstrated in the Section 2.3.3, modification of the transformation equations for substructure reanalysis is a conceptually simple process in the fixed-interface method. First, the eigenproblem solver is restarted to compute the additional fixed-fixed normal modes. Then, these new mode shapes are added to the transformation matrix and new normal DOF are computed. All computed results related to boundary DOF are unchanged by this process and thus need not be repeated.

2.2.5 Accuracy and Stability

Accuracy of results is important in two respects. Well defined modal response data is needed to accurately synthesize the higher level structures for frequency and transient analysis. Also, the quality of the displacement vectors is critical in recovery of strains and stresses within the interior of lower level substructures. Accurate stresses

require that displacement gradients be well formed. Closely tied to accuracy of the results is the numerical precision with which computations must be performed. Operations such as orthogonalization and triangulation can have a significant impact on final accuracy and the need for such operations should be considered in selecting the reduction method.

The potential for numerical instabilities in the reduction methods can be identified by examining the formulation of the methods. Typical problem areas are the divide-by-zero singularity and the linear dependence of the vectors contained in a transformation matrix.

The linear independence of the component modes in the fixed-interface transformations ensures stability of the method and accuracy has proven favorable for many problems. In fact, it is possible to obtain any level of accuracy desired simply by adjusting the number of normal DOF included in the synthesis process.

The decision to implement the fixed-interface method is supported by the above evaluation and by the role of this method as a component of several other modal synthesis techniques [1, 25, 27]. Implementation of the fixed-interface method will act as a basis for further research into modal synthesis and into other areas of structural dynamics. This study establishes the necessary first step by developing a general software system with multilevel substructuring capabilities.

2.3 Formulation of the Fixed-Interface Method

2.3.1 Basic Formulation

Consider an isolated substructure consisting of only finite elements, such as structure SPAN in Figure 1.2. The undamped, free vibration equation of motion of the substructure, partitioned to separate master (m) and slave (s) DOF, is:

$$\begin{bmatrix} K^{ss} & K^{sm} \\ K^{ms} & K^{mm} \end{bmatrix} \begin{Bmatrix} u^s \\ u^m \end{Bmatrix} - \omega_i^2 \begin{bmatrix} M^{ss} & M^{sm} \\ M^{ms} & M^{mm} \end{bmatrix} \begin{Bmatrix} u^s \\ u^m \end{Bmatrix} = (0) \qquad (2.1)$$

Master DOF are those that remain after condensation and are usually DOF at nodes on the boundary of the substructure. They are used for connectivity to adjacent substructures. The slave DOF are those that are eliminated and usually lie in the interior of the substructure. The natural frequency ω_i is that of the complete structural system, not just the isolated substructure. The presence of nonzero off-diagonal blocks $[M^{ms}]$ and $[M^{sm}]$ in Eq. (2.1) implies the use of a consistent mass formulation. When a lumped mass model is used, the mass matrix is diagonal.

The upper half of Eq. (2.1) can be expanded to

$$([K^{ss}] - \omega_i^2[M^{ss}])\{u^s\} + ([K^{sm}] - \omega_i^2[M^{sm}])\{u^m\} = (0). \qquad (2.2)$$

Solving for $\{u^s\}$ in terms of $\{u^m\}$ yields a coordinate transformation which is dependent on the unknown system vibration frequency ω_i. If the inertia forces on the slave DOF are assumed to be small compared to the static forces, the former may be neglected. Thus, the frequency dependence is eliminated and Eq. (2.2) simplifies to

$$[K^{ss}]\{u^s\} = -[K^{sm}]\{u^m\}. \tag{2.3}$$

Defining the coordinate transformation $[\varphi^c]$ from $\{u^m\}$ to $\{u^s\}$ as

$$\{u^s\} = [\varphi^c]\{u^m\} \tag{2.4}$$

$\{u^s\}$ can be eliminated from Eq. (2.3) to yield

$$[K^{ss}][\varphi^c] = -[K^{sm}]. \tag{2.5}$$

As in static condensation, $[\varphi^c]$ is evaluated by standard equation solving techniques requiring triangulation of $[K^{ss}]$ and reduction operations on the vectors in $-[K^{sm}]$. The columns of the transformation matrix $[\varphi^c]$ are known as the "static constraint modes." Physically, a static constraint mode is the displaced configuration of the slave DOF resulting from a unit displacement applied to one master DOF while all other master DOF are held fixed.

Now attention is returned to the inertia contribution of the slave DOF. If the set of master DOF is restrained from displacement, Eq. (2.1) reduces to

$$[K^{ss}]\{u^s\} - \hat{\omega}_i^2[M^{ss}]\{u^s\} = \{0\}. \tag{2.6}$$

The solution of this eigenvalue problem yields the matrix of fixed-fixed normal modes, $[\varphi^n]$, having the same order as $[K^{ss}]$ and $[M^{ss}]$. The computed vibration frequencies, $\hat{\omega}_i^2$, are those of the isolated substructure with its boundaries fixed.

The <u>complete</u> set of substructure normal modes, $[\varphi^n]$, plus the static constraint modes, $[\varphi^c]$, provide the means to transform the displacement vector $\{u\}$ from geometric coordinates to an <u>equivalent</u> set of

generalized coordinates, {q}. However, an exact transformation does not serve to reduce the order of the coordinate vector. To reduce the order of the substructure mass and stiffness matrices, the transformation to generalized coordinates is defined as

$$\{u\} = \begin{Bmatrix} u^s \\ \hline u^m \end{Bmatrix} = [T^f]\{q\} = [T^f] \begin{Bmatrix} q^n \\ \hline q^m \end{Bmatrix} . \tag{2.7}$$

The fixed-interface transformation, $[T^f]$, is derived from the static constraint modes and a <u>truncated</u> set of fixed-fixed normal modes as

$$[T_f] = \left[\begin{array}{c|c} \bar{\varphi}^n & \varphi^c \\ \hline 0 & I \end{array}\right] , \tag{2.8}$$

in which $[\bar{\varphi}^n]$ is a rectangular matrix of mode shapes selected from $[\varphi^n]$. In general, the modes corresponding to the lowest natural frequencies, $\hat{\omega}_i$, are retained in $[\bar{\varphi}^n]$. The slave displacements, $\{u^s\}$, are now dependent on both the static constraint modes and the retained normal modes of the isolated substructure. Since the full set of substructure normal modes is not used in the transformation, the generalized coordinates {q} approximately represent the geometric coordinates {u}.

Two observations regarding Eq. (2.8) are noteworthy. First, the generalized coordinate subvector, $\{q^m\}$, corresponds precisely to the master set of geometric coordinates, $\{u^m\}$. This insures geometric compatibility between adjacent substructures when the substructure equations are assembled at the next higher level of the hierarchy. Secondly, as the number of mode shapes in $[\bar{\varphi}^n]$ is reduced, the transfor-

mation shrinks to just the static constraint modes and thus, the fixed-interface method degenerates to Guyan reduction [23]. Likewise as more and more mode shapes are retained in $[\bar{\varphi}^n]$, $[T^f]$ approaches an exact coordinate transformation.

The strain and kinetic energies for the isolated substructure are given by

$$V = 1/2 \left\{ \begin{array}{c} u^s \\ \hline u^m \end{array} \right\}^T \left[\begin{array}{c|c} K^{ss} & K^{sm} \\ \hline K^{ms} & K^{mm} \end{array} \right] \left\{ \begin{array}{c} u^s \\ \hline u^m \end{array} \right\} , \text{ and} \tag{2.9a}$$

$$T = 1/2 \left\{ \begin{array}{c} \dot{u}^s \\ \hline \dot{u}^m \end{array} \right\}^T \left[\begin{array}{c|c} M^{ss} & M^{sm} \\ \hline M^{ms} & M^{mm} \end{array} \right] \left\{ \begin{array}{c} \dot{u}^s \\ \hline \dot{u}^m \end{array} \right\} ; \tag{2.9b}$$

where $\{\dot{u}\}$ is the first time derivative of $\{u\}$. The displacement and velocity vectors in Eq. (2.9) can be replaced with the generalized coordinate vectors by substitution of Eq. (2.7) and (2.8). The reduced order stiffness and mass matrices in generalized coordinates are obtained by maintaining equivalence of strain and kinetic energies between the two coordinate systems. The resulting forms are

$$[K^f] = [T^f]^T[K][T^f] = \left[\begin{array}{c|c} \hat{\omega}^2 & 0 \\ \hline 0 & K^G \end{array} \right] , \text{ and} \tag{2.10}$$

$$[M^f] = [T^f]^T[M][T^f] = \left[\begin{array}{c|c} [I] & [M^{nm}] \\ \hline [M^{mn}] & [M^G] \end{array} \right] , \text{ where} \tag{2.11}$$

$$[M^{mn}] = [M^{ms}][\bar{\varphi}^n] + [\varphi^c]^T[M^{ss}][\bar{\varphi}^n] \text{ and} \tag{2.12a}$$

$$[M^{nm}] = [M^{mn}]^T. \tag{2.12b}$$

When the substructure is composed only of elements formulated with lumped mass, the off-diagonal submatrix of equation (2.11) simplifies to

$$[M^{mn}] = [\varphi^c]^T[M^{ss}][\bar{\varphi}^n]. \tag{2.13}$$

$[K^G]$ and $[M^G]$ are the Guyan reduced stiffness and mass matrices. They take the forms

$$[K^G] = [K^{mm}] + [K^{ms}][\varphi^c] \quad \text{and} \tag{2.14}$$

$$[M^G] = [M^{mm}] + [\varphi^c]^T[M^{ss}][\varphi^c] + [\varphi^c]^T[M^{sm}] + [M^{ms}][\varphi^c]. \tag{2.15}$$

The form defined for $[K^G]$ is identical to that obtained when static condensation is applied to the stiffness in static analysis. This fact proves useful for implementation of the synthesis procedure. For the simpler case of a lumped mass formulation, Eq. (2.15) reduces to

$$[M^G] = [M^{mm}] + [\varphi^c]^T[M^{ss}][\varphi^c]. \tag{2.16}$$

The identity submatrix in $[M^f]$ and the submatrix $[\hat{\omega}_n^2]$ in $[K^f]$ result from the orthonormality of the mode shapes in $[\bar{\varphi}^n]$. $[\hat{\omega}^2]$ is a diagonal matrix of natural frequencies corresponding the the modes retained in $[\bar{\varphi}^n]$.

The normal coordinates are coupled to the geometric DOF only in the reduced mass matrix (submatrices $[M^{mn}]$ and $[M^{nm}]$). The off-diagonal submatrices of $[K^f]$ are null as a consequence of the equation development.

Regardless of which mass matrix formulation is used, consistent or lumped, the reduced mass submatrix, $[M^G]$, is fully populated. The computational advantage of a lumped mass formulation is therefore limited to reduction of the lowest level substructures in the hierarchy.

When time-dependent loads are applied to the slave DOF, they too must be transformed to generalized coordinates. If the substructure is subjected to an arbitrary virtual displacement, $\{\delta u\}$, the work done by the substructure forces $\{P\}$ is

$$\delta W = \{\delta u\}^T \{P\}. \tag{2.17}$$

The condensed forces, $\{F\}$, applied to the generalized coordinates must do the same work during a virtual displacement consistent with $\{\delta u\}$, thus

$$\{\delta q\}^T \{F\} = \{\delta u\}^T \{P\}. \tag{2.18}$$

Substituting Eq. (2.7), the condensed force vector becomes

$$\{F\} = [T^f]^T \{P\}. \tag{2.19}$$

The stiffness, mass, and loads for each substructure are partitioned and condensed. Assembly of both the reduced substructure mass and stiffness into the next higher level follows the standard procedure for element assembly [10]. Displacement compatibility between adjacent substructures is automatically insured by the use of the master DOF as generalized coordinates. Although assembly of the reduced substructure stiffness and mass is routine, an illustration of the final matrices is useful. For an assembly of "r" substructures

$$[\overset{*}{K}] = \begin{bmatrix} \hat{\omega}_1^2 & 0 & \cdots & 0 & 0 \\ 0 & \hat{\omega}_2^2 & & & 0 \\ \vdots & & \ddots & & \vdots \\ 0 & & & \hat{\omega}_r^2 & 0 \\ 0 & 0 & \cdots & 0 & \overset{*G}{K} \end{bmatrix} ; \qquad [\overset{*}{M}] = \begin{bmatrix} I & 0 & \cdots & 0 & M_1^{nm} \\ 0 & I & & 0 & M_2^{nm} \\ \vdots & & \ddots & & \vdots \\ 0 & 0 & & I & M_r^{nm} \\ M_1^{mn} & M_2^{mn} & \cdots & M_r^{mn} & \overset{*G}{M} \end{bmatrix}$$

(2.20a) (2.20b)

The master DOF from the various substructures are coupled only in the submatrices $[\overset{*G}{K}]$ and $[\overset{*G}{M}]$, the assembled Guyan stiffness and mass.

The synthesis process for one level of substructuring is now complete. After a free-vibration analysis has been performed for the synthesized structure, it may be desirable to recover the portion of the system mode shapes contained within the condensed substructures. This is achieved by applying Eq. (2.7) to that portion of the system mode shape associated with the generalized DOF from a particular substructure.

In summary, the fixed-interface method employs static constraint modes and a truncated set of fixed-fixed normal modes to achieve a reduction in the order of the substructures stiffness and mass. Geometric coordinates at internal boundaries are retained in the set of generalized coordinates to insure displacement compatibility between substructures.

2.3.2 Extension to Multilevel Substructuring

The fixed-interface method is extended to multilevel substructured modeling in the following manner. Referring to the terminology of section 1.2, assume that all substructures at level "i" have been assembled

either from finite elements or level "i+1" substructures (or both). The level "i-1" substructures are defined by selecting master and slave DOF for each substructure at level "i", condensing these substructures using Eq. (2.10) and Eq. (2.11), and assembling as illustrated in Eq. (2.20a,b).

A significant difference in the procedure for multilevel substructured models from that of the preceding section is the selection of master and slave DOF. As previously mentioned, master DOF are usually selected to lie along substructure boundaries and slaves are chosen as the remaining DOF. For the normal DOF which exist as a result of the synthesis of condensed substructures, no physical basis exists upon which to make this selection. Conceptually, the normal DOF in the assembled model could be identified as either master or slave DOF.

For this study, the following procedure is adopted. Since the equations of constraint that link adjacent substructures are written only in terms of the substructure boundary (geometric) DOF, the normal (generalized) DOF for each substructure are grouped with the interior DOF in the set of slaves.

As an example, consider structure "A" which is assembled from two condensed substructures, "B" and "C". The assembled stiffness and mass matrices for structure "A" are illustrated in Figure 2.1. The matrices are partitioned into five zones as indicated. Zone I and II contain the normal DOF from substructures "B" and "C" respectively. The identity matrices in $[M_A]$ and the diagonal blocks of substructure frequencies in $[K_A]$ are fully contained within the individual zones. This illustrates that normal DOF from one substructure are not coupled with those from

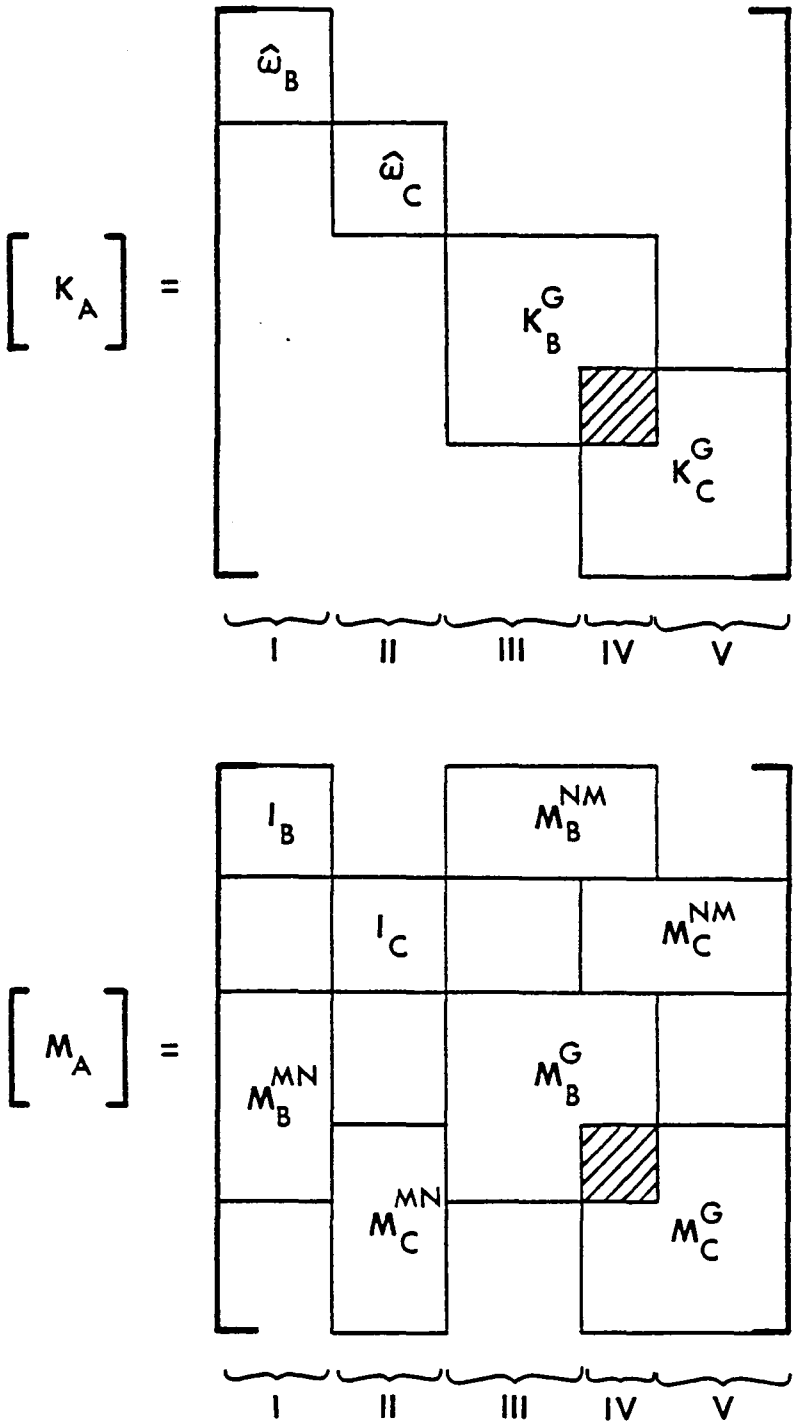


Figure 2.1. Substructure Equation Assembly .

adjacent substructures. The boundary DOF of substructure "B" occupy zones III and IV while zones IV and V contain boundary DOF from substructure "C". Clearly zone IV represents the boundary DOF common to "B" and "C". The DOF in this zone are linked to enforce displacement compatiblity between the substructures.

In one-level substructured models, this representation of structure "A" would form the highest level structure and the synthesis process would be complete. In multilevel substructured models, structure "A" is partitioned into its own master and slave DOF and then condensed. As mentioned above, master DOF are usually selected as those DOF on substructure boundaries. In this respect, the master DOF for structure "A" are selected from zones III, IV, and V. The remaining DOF in these three zones, along with all generalized DOF in zones I and II are grouped as slave DOF. The synthesized stiffness and mass matrices resulting from condensation of structure "A" are identical in form to the stiffness and mass matrices from any other condensed structure; see Eq. (2.10) and (2.11). An evaluation of the impact of the above master/slave selection procedure for multilevel substructured models remains a topic for future study.

2.3.3 Substructure Reanalysis

When modal synthesis is used to condense the substructures in a complex structural model, analysts will always question the accuracy of the reduction and thus the quality of the final results. Substructure reanalysis is the most obvious approach to verifying the representation of an individual substructure. In the fixed-interface method, substructure reanalysis is achieved simply by adding more normal DOF to the

condensed substructures in question. Many of the computations performed in the initial reduction need not be repeated during reanalysis. Consequently, reanalysis is performed with some degree of efficiency when computed results are retained after completion of the initial analysis.

The first step in substructure reanalysis is to determine which additional normal DOF are to be retained in the condensed substructure. If sufficient fixed-fixed normal modes are not available for addition to the transformation $[T^f]$, the eigenproblem solver is restarted to compute the required frequencies and mode shapes. Existing fixed-fixed normal modes are not recomputed.

After the additional normal DOF for the substructure are computed, the condensed stiffness and mass matrices are assembled. Referring to equations (2.10) and (2.11), the Guyan reduced stiffness and mass submatrices, $[K^G]$ and $[M^G]$, remain unchanged since the normal DOF do not influence the static constraint modes. The only computations required are those needed to expand the number of columns in the off-diagonal mass submatrix, $[M^{mn}]$. These new columns are needed for the additional substructure normal DOF. Matrices $[\hat{\omega}^2]$ in $[K^f]$ and $[I]$ in $[M^f]$ are similarly expanded.

Savings in the assembly of "reanalyzed" substructures are also possible. Using the example presented in the previous section, suppose that additional normal DOF have been added to substructures "B" and "C." When the stiffness and mass matrices for structure "A" are reassembled, only zones I and II need to be expanded (Figure 2.1). Since the Guyan stiffness and mass submatrices for both "B" and "C" do not change during

reanalysis, their assembly into structure "A" is also unchanged. Thus zones III, IV, and V are not altered, saving measurable time in structure assembly.

While the foregoing procedure is conceptually simple, implementation of reanalysis capabilities in a general software system presents some special problems not yet considered. Details of this implementation are presented in Section 4.8.

CHAPTER 3 -- SOFTWARE DEVELOPMENT ENVIRONMENT

3.1 General

The fixed-interface method provides a theoretical basis to perform dynamic analysis of multilevel substructured FEM models. Design and implementation of the associated software for general-purpose analysis makes the procedure accessible to researchers and designers. Finite element researchers typically focus on developing and improving numerical algorithms, not on the design and implementation of sophisticated engineering software. Software for these researchers is implemented only to demonstrate the viability of the numerical method for a limited class of problems. As a consequence, the software tends to be deficient in the areas of user-interface, resource management, and generality.

The programming capabilities needed to overcome these deficiencies are not supported by standard algorithmic languages (e.g. FORTRAN-77, C, Pascal). A software developer who wishes to use hierarchial data structures, for example, is required to devise his own data management capabilities. This task typically results in complex sequences of procedure calls from the processing routines in order to locate or create the necessary data tables. For advanced applications, such as substructured modeling and nonlinear analysis, implementation of the numerical procedure becomes a trivial task compared to the "bookkeeping" procedures required to drive the crude data management routines.

One solution to this problem is the use of an "executive" system to support and manage computing resources: memory, secondary storage, data transfers between the two, and user-interface. The POLO system [42, 43] provides the necessary support. The software developed during

this study relies heavily on the POLO executive. The software development tools within POLO enable the areas of engineering mechanics, numerical methods, and computer science to be effectively synthesized into a functioning software system having considerable generality. The remainder of this chapter briefly describes the components of POLO and its influence on the software developed in this study. For additional details on the POLO executive and on the concept of software virtual machines, see [16] and [17].

3.2 The POLO Executive

POLO does not directly solve engineering problems. Rather it supports programming activities common to most engineering applications: POL translation, data structure definition, data base and memory management during execution, and logical control and integration of application subsystems. A specific application program, or subsystem, which runs under the control of POLO is needed to solve the engineering problem. The existing finite element subsystem for POLO, named POLO-FINITE, has been adopted as the starting point for the software developed in this study.

POLO supports engineering software applications during the development phase and during execution of the application program (also known as "run-time"). During development, POLO provides languages to define data structures, to symbolically access the data, and to control the sequence of operatons on data required for the particular application. At run-time, POLO support routines perform data base and memory management, translate POL input, and execute the processing routines. At

program termination, POLO automatically secures all data bases for subsequent analysis restart.

POLO provides compilers and execution processors for two higher level languages: a data definition language (DDL) and a host language (HL). These two languages and an algorithmic language (FORTRAN-77) combine to define the development environment (Figure 3.1). The individual components of this environment and their inter-relationships are discussed in the following sections. Section 3.6 describes the run-time configuration of a POLO application program. The structure of POLO-FINITE as a FEM application program is presented in the next chapter. A more complete discussion of POLO-FINITE, including system performance, nonlinear analysis capabilities, and element and material model libraries, can be found elsewhere [16, 18, 43].

3.3 Data Definition Language

The development of a POLO subsystem centers on the structure of the logical data space. Data structures in the POLO environment are primarily of the hierarchical type. Other data structures, including network and relational, may be defined using basic hierarchical tables with additional pointer manipulation by the application subsystem. Data structures are described to POLO with the data definition language (DDL). As shown in Figure 3.1, the developer's data definition is compiled into an internal form by the DDL compiler. The resulting form of the data definition resides in the DDL library. The DDL library contains the logical definition of and the relationships among all data structures defined for the application program. This library is later accessed by the host language (HL) development processors to interpret

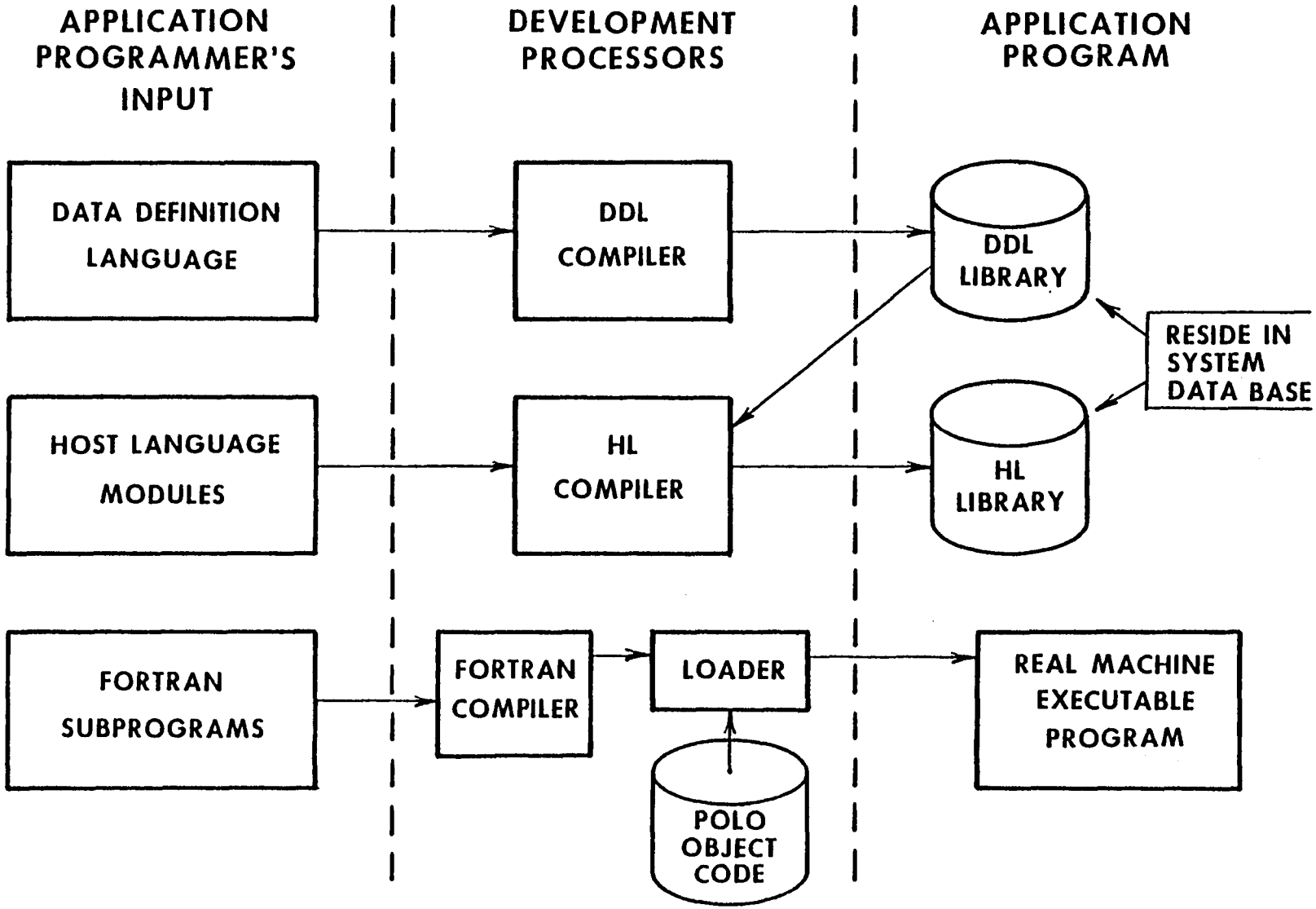


Figure 3.1. POLO Development Environment

data references made in the HL programs. At run-time, the data definition is used to map the logical data format onto a physical medium (direct-access disk file) for the storage of problem data.

Figure 3.2 contains a sample data hierarchy defined for the dynamic analysis systems. In this example the stiffness, mass, and frequency analysis results are all stored in a table named COEFFICIENTS which has its rows labelled (or named) and is one column wide. The COEFFICIENTS table actually resides in a higher level table, ELEMENTS, which contains other relevant structure data: nodal coordinates, element incidences, constraints, loads, etc. The DDL for the sample data structure is presented below.

```
TABLE ELEMENTS LABELLED GROUPING 25
        .
        .
        .
   TABLE COEFFICIENTS LABELLED 1
      NNODE    INTEGER
      NROW     INTEGER
      NCOL     INTEGER
      TABLE STIFFNESS LABELLED NNODE
         KLOW        INTEGER
         NUMBLOCKS   INTEGER
         TABLE ROWS ARRAY REAL NUMBLOCKS NROW NCOL
         END OF TABLE
      TABLE MASS       LABELLED NNODE
         MLOW        INTEGER
         NUMBLOCKS   INTEGER
         TABLE ROWS ARRAY REAL NUMBLOCKS NROW NCOL
         END OF TABLE
      TABLE LUMPEDMASS SET REAL NNODE NROW
      TABLE FIXEDMODES LABELLED GROUPING 50
         FREQUENCY REAL
         TABLE SHAPES  SET REAL NNODE NROW
         END OF TABLE
      TABLE FREEMODES  LABELLED GROUPING 50
         FREQUENCY REAL
         TABLE SHAPES  SET REAL NNODE NROW
         END OF TABLE
      END OF TABLE
         .
         .
         .
```

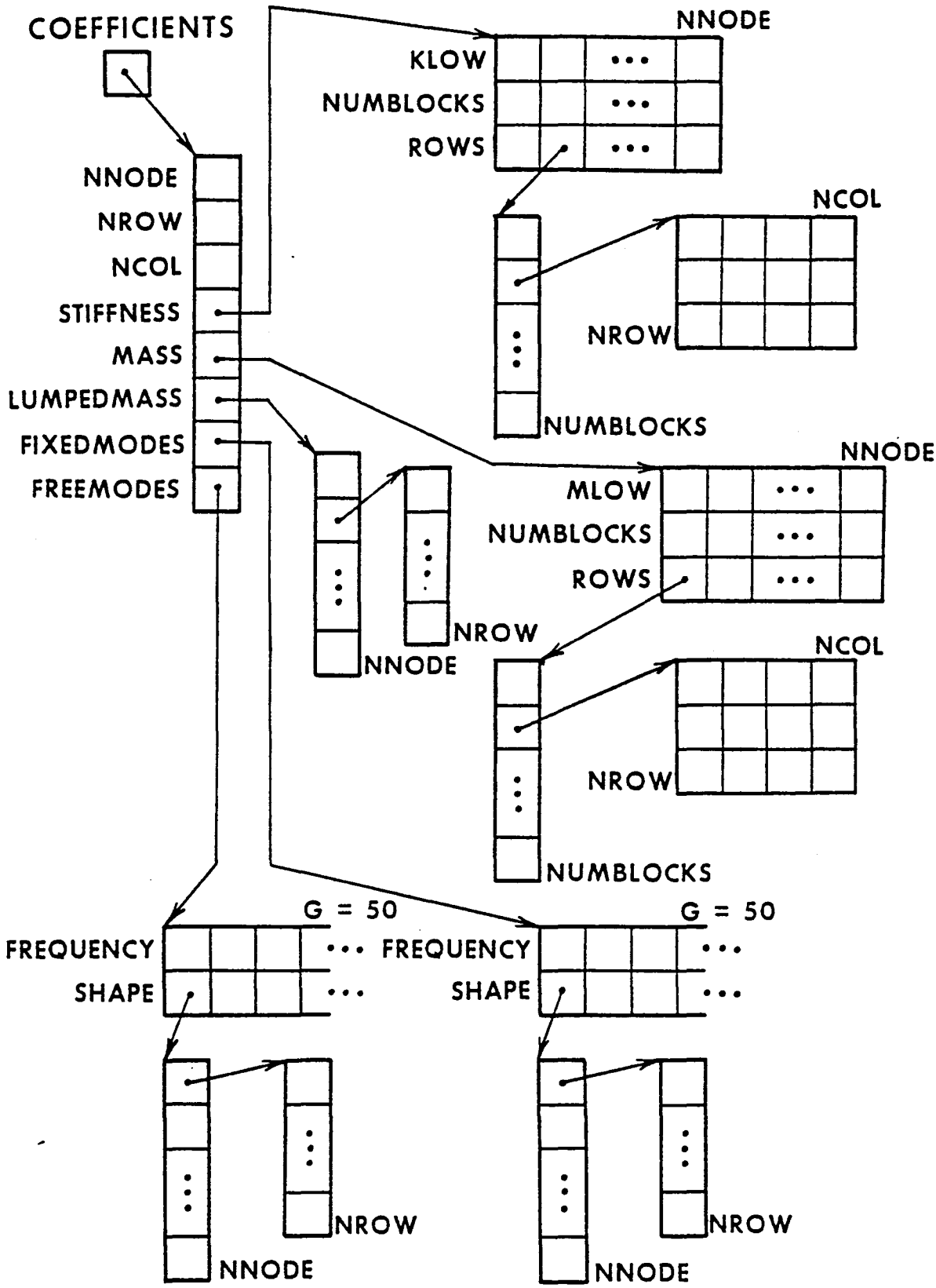


Figure 3.2. Sample Data Structure

The first three rows in table COEFFICIENTS are scalar entries. The values are used to define sizes of lower level tables. The fourth row of this table begins the definition of a labelled table named STIFFNESS. This "table within a table" is defined simply in the DDL as shown. Definition of other rows in COEFFICIENTS is temporarily suspended until table STIFFNESS is fully specified. After the three rows of the STIFFNESS table are described, the END OF TABLE statement indicates that the statements to follow define other rows of the COEFFICIENTS table.

For a consistent mass formulation, the mass matrix has the same banding as the stiffness matrix. Thus the MASS table has a hierarchy which is identical in structure to the STIFFNESS table. A different data structure is appropriate for a lumped mass formulation in which DOF coupling does not exist. The table LUMPEDMASS defines the values of mass that reside at each DOF of every structure node. While both mass tables (consistent and lumped) are specified for each structure, only the one table that corresponds to the selected mass formulation for the structure is created in the data base at run-time.

In a similar manner, two types of frequency analysis tables, FIXEDMODES and FREEMODES, are defined. While both tables are defined for any given structure, only the appropriate one is created to store the results of the analysis. The FIXEDMODES table stores the fixed-fixed frequencies and mode shapes for condensed substructures during modal synthesis, while the FREEMODES table contains analysis results for a free-vibration frequency analysis of the highest level structure.

In static analysis, only the STIFFNESS table is created at run-time. In the problem data base, the rows of the COEFFICIENTS table

corresponding to the mass and mode shape tables then contain pointer values of zero, indicating that the tables have not been created.

When a table is first referenced at run-time, it is created according to the sizes defined in the DDL. If any of the sizing parameters are variables, the data manager creates the table using the current value of the variables. The sizing variables can then be changed during execution of the application program so other tables can be created to different sizes as required.

The FIXEDMODES and FREEMODES tables are slightly different from the other labelled tables in the data structure. These two tables are known as "grouped" tables and have a grouping factor of 50 (an arbitrary choice). These tables are initially created with 50 columns. As additional columns of the table are needed, they are created in groups of 50 each. The groups of 50 columns are not necessarily contiguous in the database. The COEFFICIENTS, STIFFNESS, and MASS tables are not grouped. All columns required for each of these tables are allocated contiguously in the database on the first reference to the table.

The data definition listed above is just a small part of the data definition used in POLO-FINITE. Additional details regarding the specific data structures developed in this study are presented in Chapter 4.

3.4 Host Language

The second component of the POLO development environment is the host language. After the developer has defined the data structures, host language programs are written to drive execution of the application subsystems. An HL program performs three primary functions: POL input

translation, execution of FORTRAN support routines, and execution of other POLO subsystems. These functions are directed in HL command statements that have a basic IF-THEN syntax.

The syntax of a HL command statement takes the following form:

<label> <logical test> <action list> <transfer destinations>.

The label is optional and serves the same purpose as a statement label in FORTRAN. The logical test is evaluated to determine whether or not the action list will be executed. If the result of the logical test is false, the actions are skipped and the "false transfer of control" is taken. If the logical test is true, the actions in the list are executed and the "true transfer" is performed. The actions executed by the HL processor typically involve numerical computations that are efficiently performed in the FORTRAN support routines (matrix multiplies, etc.).

It may not always be appropriate to perform a logical test prior to executing a list of actions. When this is the case, a dummy test, *EXECUTE, is performed. The result of this test is always true and the action list is executed. A situation in which a dummy true-test is appropriate might be the execution of initialization routines at the entry point to a subsystem.

Data references may be associated with each action in the action list. A data reference is a symbolic reference to tables within the hierarchy as defined in the DDL. An example of a data reference into the hierarchy defined in the previous section is the following:

```
/STRUCTURE/ELEMENTS( COEFFICIENTS, STCOL, FREEMODES, 1, SHAPES,
                     IMODE, INODE, 1 )
```

This data reference accesses the free-vibration mode shape data for a particular vibration mode. The reference begins by identifying the data base that contains the required data (the name is enclosed in "/ /"). Then starting with the name of a table defined at the highest level, the hierarchy is symbolically traversed. STCOL is a variable that contains the column number in the ELEMENTS table which contains data for the desired structure; variable IMODE contains the column number in the FREEMODES table that identifies the individual mode shape of interest; variable INODE contains the structure node number required. The traversed rows of the labelled tables are referenced by name (COEFFICIENTS, FREEMODES, and SHAPES). Lower levels of the data hierarchy are reached by appending additional subscripts to the reference.

A complete example of an HL command statement is given by:

```
LUMP_MASS  *COMPARE( MASTYP, 1 ),
               MOVEDATA( SCRTCH, /SOLVER/STRUCTURE(LUMPEDMASS,ICOL,1,1)),
               JACOBI( /SOLVER/STRUCTURE( STIFFNESS,ICOL,1,1 )),
               GO TO SORT_RESULTS, CONSIS_MASS
```

This command statement is taken from the HL program which performs eigenproblem solution by the generalized Jacobi method. LUMP_MASS is the statement label used as a transfer destination. In this case the logical test is *COMPARE in which the variable MASTYP is compared to the integer 1 for equality. A MASTYP of 1 implies that the mass formulation for the structure is lumped. If the result of the test is true, two subsystem actions are executed. The MOVEDATA action copies the contents of the LUMPEDMASS table from the SOLVER data base to the array SCRTCH. If the data reference does not include a data base specification (ex. /SOLVER/) the data item is a variable in COMMON. Action JACOBI performs the eigenproblem solution using the STIFFNESS table from the SOLVER data

base and the mass data previously placed into COMMON by MOVEDATA. After the actions have been executed, control is transferred to the statement with the label SORT_RESULTS. If the result of the logical test is false, the actions are not executed and control is transferred to the statement labelled CONSIS_MASS.

As implied in the preceeding example, the HL programs and the FORTRAN actions communicate through a COMMON area. When a particular subsystem action is invoked by the HL program, the corresponding FORTRAN subroutine is identified by variables in COMMON. Also, when data from a data base is needed for execution of a subsystem action, the data manager moves that data into COMMON. These two methods of subsystem communication require that COMMON be divided into two sections. The first section is the static COMMON area. This portion of COMMON contains variables required throughout execution of a subsystem (MASTYP and SCRTCH in the previous example). The second portion of COMMON contains the dynamic pool which is partitioned into equally-sized pages. The data manager places the data which is referenced by an action call into the dynamic pool. When a data reference is resolved at run-time, the data manager moves the data from the application data bases to the dynamic pool. Paging of existing data in the pool to make room for new data is handled automatically.

Each HL program contains an action list which establishes the relationship between action names referenced in the HL and the corresponding FORTRAN subroutines. A portion of the action list related to the previous example takes the form:

```
ACTIONS TYPE 33
          .
          .
          .
  JACOBI      7
          .
          .
          .
END OF ACTIONS
```

The subsystem number 33 and the action number 7 are placed in static COMMON to identify the FORTRAN subsystem and the subprogram which correspond to the JACOBI action. Frequently used actions, such as MOVEDATA, do not appear in the action list for the application subsystem. POLO supports these actions as an integral part of the HL in the same manner as FORTRAN provides the intrinsic functions: SIN, COS, etc.

The completed HL programs are compiled by the HL compiler and the object code is stored in the HL object library (Figure 3.1). This library is also a part of the system data base. The HL compiler refers to the DDL library to generate appropriate instructions as the data references are resolved. The HL compiler checks each HL program for command syntax errors and data references which are inconsistent with the DDL. The subsystem developer receives appropriate messages when the compiler detects these coding errors. When subsystem development is complete, the object form of the HL programs act as instructions for the POLO "virtual processor." The next section contains a brief description of the virtual processor.

3.5 FORTRAN Processing Routines

As mentioned above, POLO is not capable of solving engineering problems by itself. The set of actions available to developers is limited to those procedures needed for data management, POL translation, logical control, and other utility operations (ex: MOVEDATA). Numerical operations such as matrix addition, multiplication, and triangulation are not supported by the HL. Unacceptable overhead is incurred if operations of this type (requiring loop indexing and array subscripting) are coded in the HL. Instead, FORTRAN subroutines are written to perform the numerical computations. The generalized Jacobi method referenced in the previous section is a good example. Once the data manager places the necessary data in COMMON as a result of an HL reference, all numerical computations are efficiently performed in one FORTRAN subroutine.

The FORTRAN subprograms are compiled with the FORTRAN compiler for the host computer system. The resulting object code is combined with the object code library of POLO (also compiled FORTRAN) and loaded into a single executable program (Figure 3.1). This real machine program and the system data base comprise the final application program.

A distinction is made here between the instructions generated by the FORTRAN compiler and those generated by the POLO compilers (DDL and HL). The FORTRAN compiler generates "real-machine" instructions which are executed by the hardware processor. The POLO compilers generate "virtual-machine" instructions which are interpreted by the POLO virtual processor. A virtual instruction consists of an action to be performed and a description of the data necessary to perform that action. Execution of a virtual-machine instruction by the POLO virtual processor

typically results in 5-10 FORTRAN subroutine calls to the data base and memory manager, followed by a reference to the action subprogram.

To demonstrate the link between HL programs and the FORTRAN subroutines, the JACOBI action is examined in more detail. Action JACOBI is defined in the HL as action number 7 of subsystem 33. When the POLO virtual processor interprets an instruction to execute JACOBI, it places the integer 7 in a COMMON variable. The data reference associated with the JACOBI action is resolved and the corresponding data is moved to the dynamic pool if it is not already there. The location of the data and the dimensions of the table from which it is obtained are also stored in the COMMON area. A call to SUBROUTINE TGTY33 (subsystem 33) is issued by POLO and control is thus transfered to the application subsystem. The first few lines of the subsystem take the form:

```
      SUBROUTINE TGTY33
      COMMON /TGUSER/ RPOOL(1), ...
      COMMON /PARAM/  IACTION, LOC1, LOC2, ...
            .
            .
            .
      GO TO( 100, 200, 300, ...  ), IACTON
            .
            .
            .
  700 CALL JACOBI( RPOOL(LOC1), SCRTCH, ... )
      RETURN
            .
            .
            .
      END
```

The action number to execute is identified by IACTON. Subroutine JACOBI is passed the stiffness matrix (which starts at location LOC1 in the dynamic pool) and the lumped mass vector (stored in COMMON variable SCRTCH by a prior call to action MOVEDATA). Array dimensions are also passed to subroutine JACOBI so the data located in vector RPOOL can be

treated in its appropriate form (vector, matrix, or three-dimensional array). The first few lines of subroutine JACOBI are:

```
SUBROUTINE JACOBI( STIFF, XMASS, NRSTIF, NCSTIF )
DIMENSION STIFF( NRSTIF, 1 ), XMASS( 1 )
      .
      .
      .
```

3.6 Run-Time Configuration

The integration of POLO and the application subsystems into a single executable program is illustrated in Figure 3.3. The POLO virtual processor is the highest level driver and takes its instructions from the compiled HL programs in the system data base. The virtual processor drives the POL scanner, the data and memory managers, and the application subsystems. After program initialization, the virtual processor is instructed to read POL input from the current input device (the user's terminal during interactive execution or a sequential disk file during batch execution). The user's input is translated to fixed format by the POL scanner and is placed at the top of the COMMON area. Input is read one line at a time and acted upon as required.

The virtual processor calls the application subsystem after the data manager has resolved the data reference and the memory manager has placed the necessary data in COMMON. An application subsystem is composed of an executive routine (ex. SUBROUTINE TGTY33) and a number of lower level subprograms (ex. SUBROUTINE JACOBI). The application subsystem has access to only the data in the COMMON area. The memory manager controls all data transfers between the application data bases and COMMON.

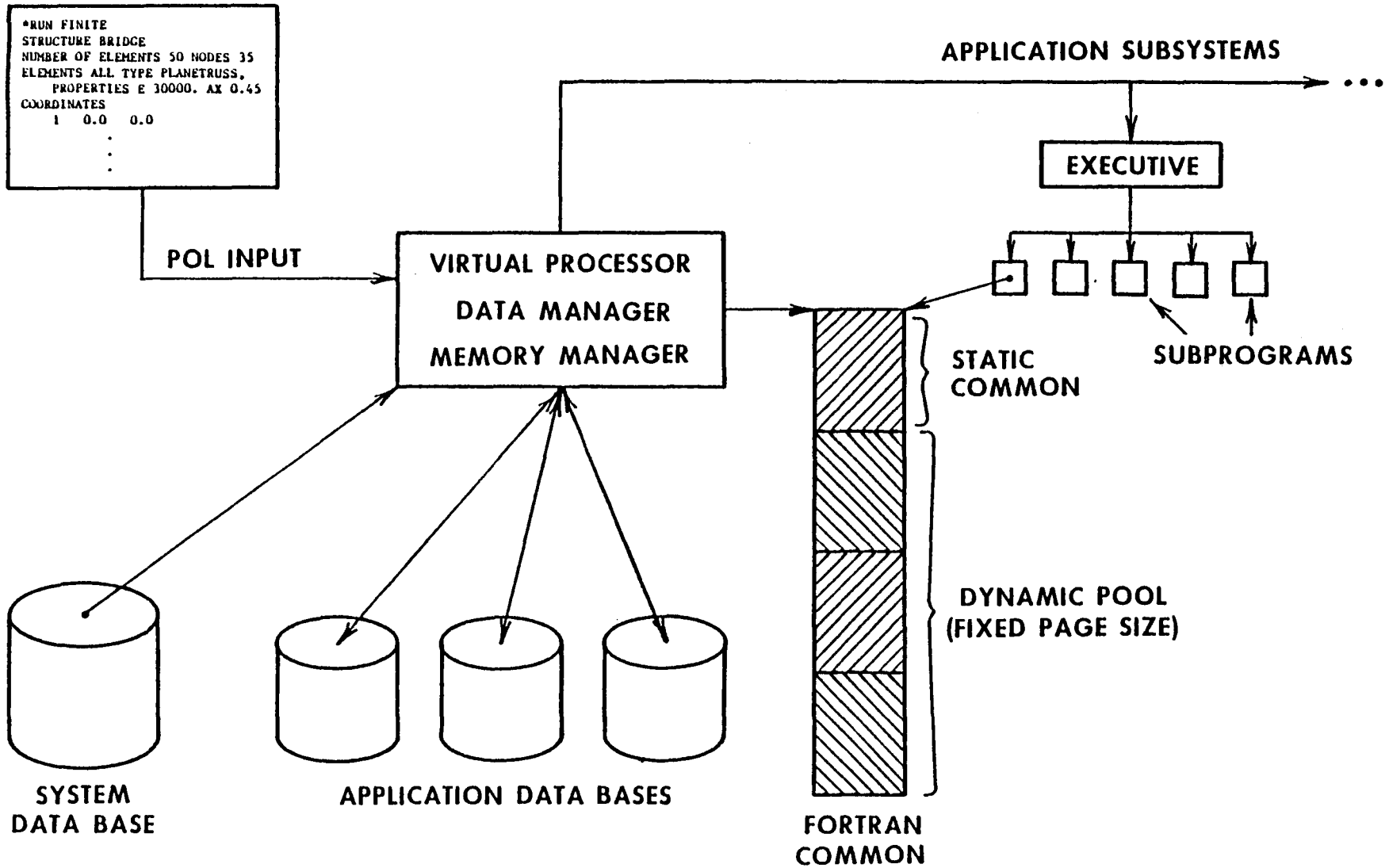


Figure 3.3. POLO Run-Time Configuration

Figure 3.4 more clearly illustrates the functions of the data and memory managers. Instructions which describe the data reference for an action are stored in the object code of the HL. In order to make the requested data physically present in memory, the data reference instructions are passed to the data manager. The data manager converts the instructions into a virtual address which identifies the file, page (record), and position on the page that the requested data occupies in the application data base. If the data does not currently exist in the data base, the table is created. The memory manager converts the virtual address to a physical address. If the data is currently in COMMON, the pool subscript (LOC1 in the above example) is returned to the data manager. If the data is not in COMMON, the memory manager makes room for the new data by swapping an existing data page back to its disk file and then reading the new page into the available memory location. The pool subscript is then returned to the data manager. Since all pages in the data base are the same size, fragmentation of the dynamic pool during execution does not occur.

With all data references for a particular action resolved, the data manager places the physical address and sizing information for each data item into the static portion of COMMON. Control is returned to the virtual processor which calls the application subsystem. At termination of the program, the data and memory managers write all data in the dynamic pool back to the application data bases. The data bases are thus preserved for subsequent analysis restart.

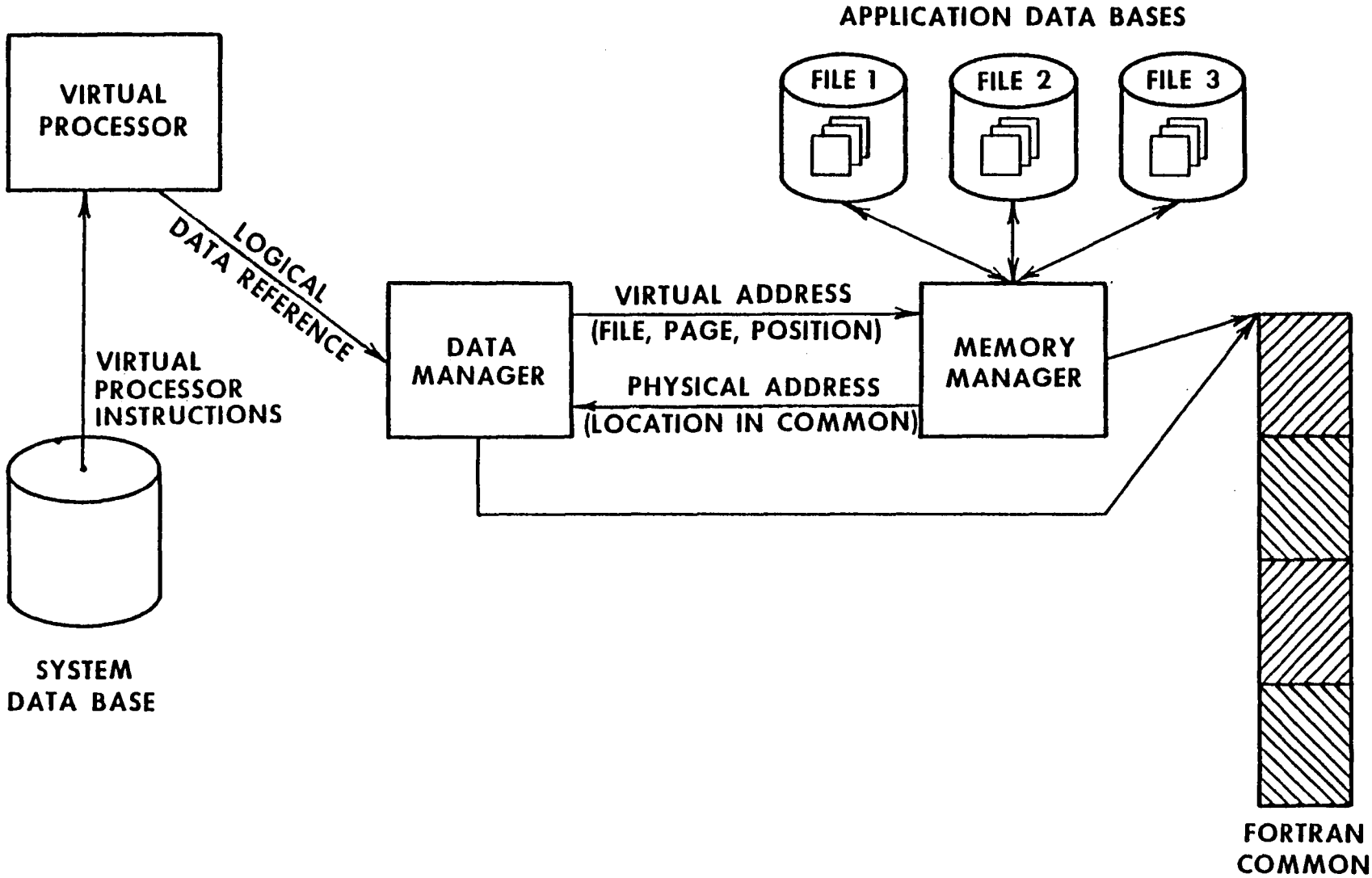


Figure 3.4. Resolution of Logical Data References

CHAPTER 4 -- SOFTWARE DESIGN AND IMPLEMENTATION

4.1 General

Most finite element researchers are concerned with only the formulation and resulting behavior of numerical algorithms. Software implementation is generally given only cursory attention. Fortunately, software engineering is becoming more accepted as a research topic in itself. Real-world considerations for program size, development and maintenance costs, user interface, execution time, and data management have led to the formal study of software development techniques. This chapter discusses the approach to software development followed in this study. Since a complete review of the entire implementation is neither desirable nor practical, only a few specific examples of data structures and computational procedures are presented for illustration.

The design and implementation of the software to perform dynamic analysis relied heavily on the support of the POLO executive and the existing organization of FINITE. Development of the data structures and computational subsystems followed the same techniques that were used in the initial development of POLO-FINITE. This approach assured a compatibility between existing analytical functions and new procedures which served to minimize the need for new code. Most of the existing subsystems in FINITE, such as the output processor, required only minor modification. When computational procedures unique to dynamic analysis required the development of new subsystems, they were developed so that existing subsystems could be utilized wherever possible. For example, the new subsystem for solution of the eigenproblem uses the triangulation and load-pass subsystems already available for static analysis.

This chapter summarizes the design and implementation aspects of the software developed for this study. The next section contains a description of FINITE, illustrating the logical structure of the individual subsystems and databases for both static and dynamic analysis. The POL that facilitates the new structural modeling and analysis capabilities is then presented. New data structures for dynamic analysis are described. Control of the solution process is presented in detail. The significant solution algorithms implemented for this study are discussed followed by a review of the restart and reanalysis procedures.

4.2 FINITE System Organization

FINITE is the POLO application program which supports finite element analysis. The FINITE system consists of a number of subsystems (each containing an HL program and a set of FORTRAN subprograms), three logical databases, and the POLO executive. The organization of the FINITE system is illustrated in Figure 4.1. POLO lies at the center of the system and controls execution of the various subsystems along with all data flow to and from the databases. Obviously, some functional dependencies must exist among the FINITE subsystems in order to solve finite element problems. As indicated in Figure 4.1, these dependencies are transparent to POLO. POLO treats each subsystem (and each database) as an independent unit.

The following section describes some of the FINITE subsystems. The function of the individual subsystems and their relationships with each other are discussed. A general description of the three databases is then given. Specific data structures used in problem solution are

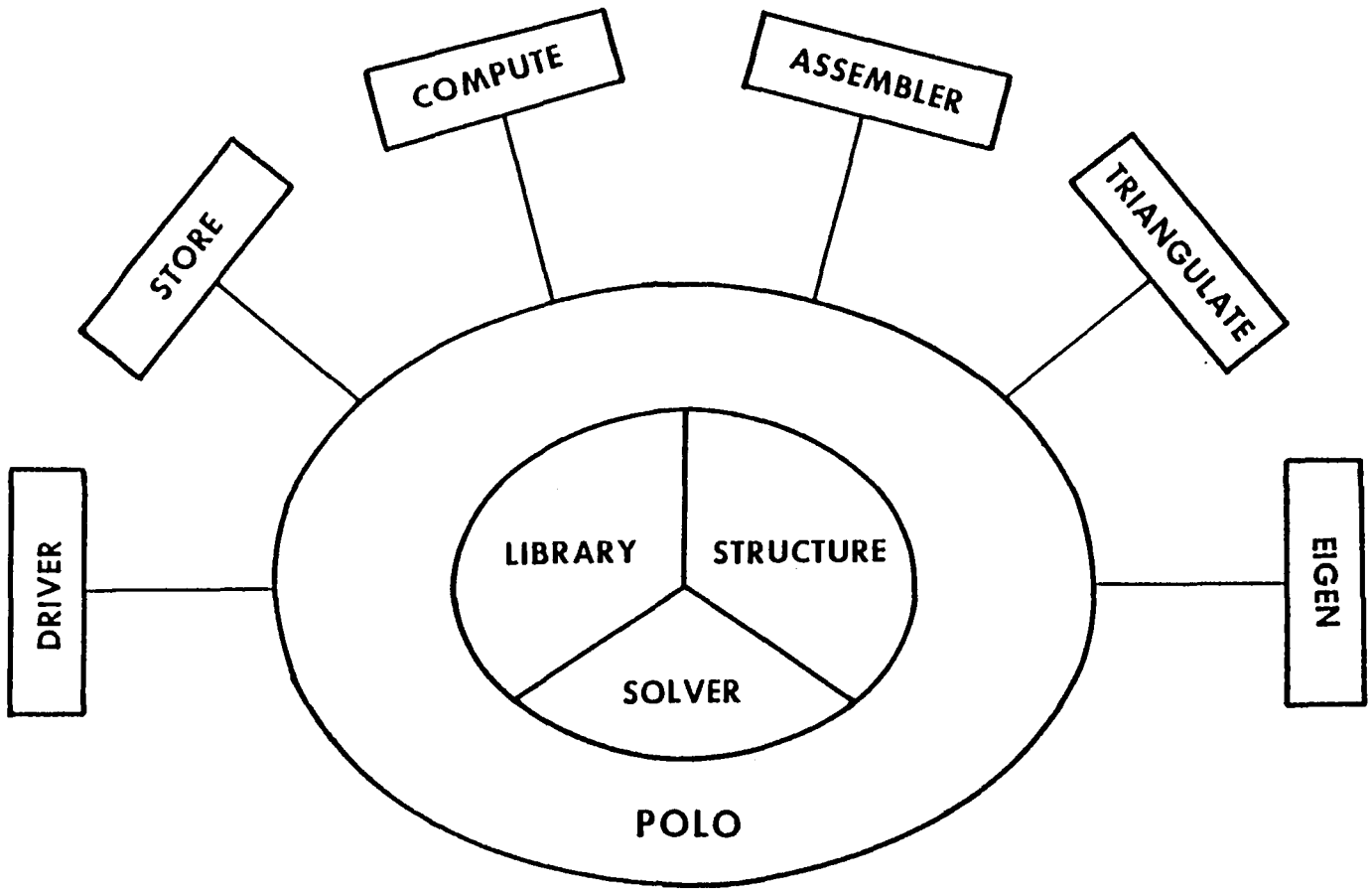


Figure 4.1. FINITE System Organization

presented later in the chapter. Finally, the technique for interfacing the FINITE subsystems is presented.

4.2.1 Organization of FINITE Subsystems

Figure 4.2 illustrates the functional dependencies among the FINITE subsystems. Subsystem DRIVER is the highest level subsystem in FINITE and is the entry point for the command: *RUN FINITE. This subsystem ensures that the three databases exist and processes the highest level user input commands. Through an internal POLO "RUN" command in its HL, subsystem DRIVER invokes one of three subsystems: LIBRARY, STORE, or COMPUTE to continue processing user input.

Subsystem LIBRARY is used by system developers to maintain tables that define all finite elements and nonlinear material models. Element tables contain information on the characteristics of each element, such as the number of nodes, the types of DOF at each node, user-definable properties, and possible mass and nonlinear formulations. Material model tables describe the characteristics of the material, such as initial material properties, the type of stress-strain or load-deformation functions that may be used, and material hardening rules. Subsystem LIBRARY is essentially an editor which maintains the LIBRARY database. The function of subsystem LIBRARY is transparent to the user who is not involved with system development.

Subsystem STORE translates user input that defines the characteristics of a structural model for subsequent analysis. Structural geometry, loads, constraints, element selections, and solution procedures are all translated by STORE. This information is checked for consistency and placed into the STRUCTURE database.

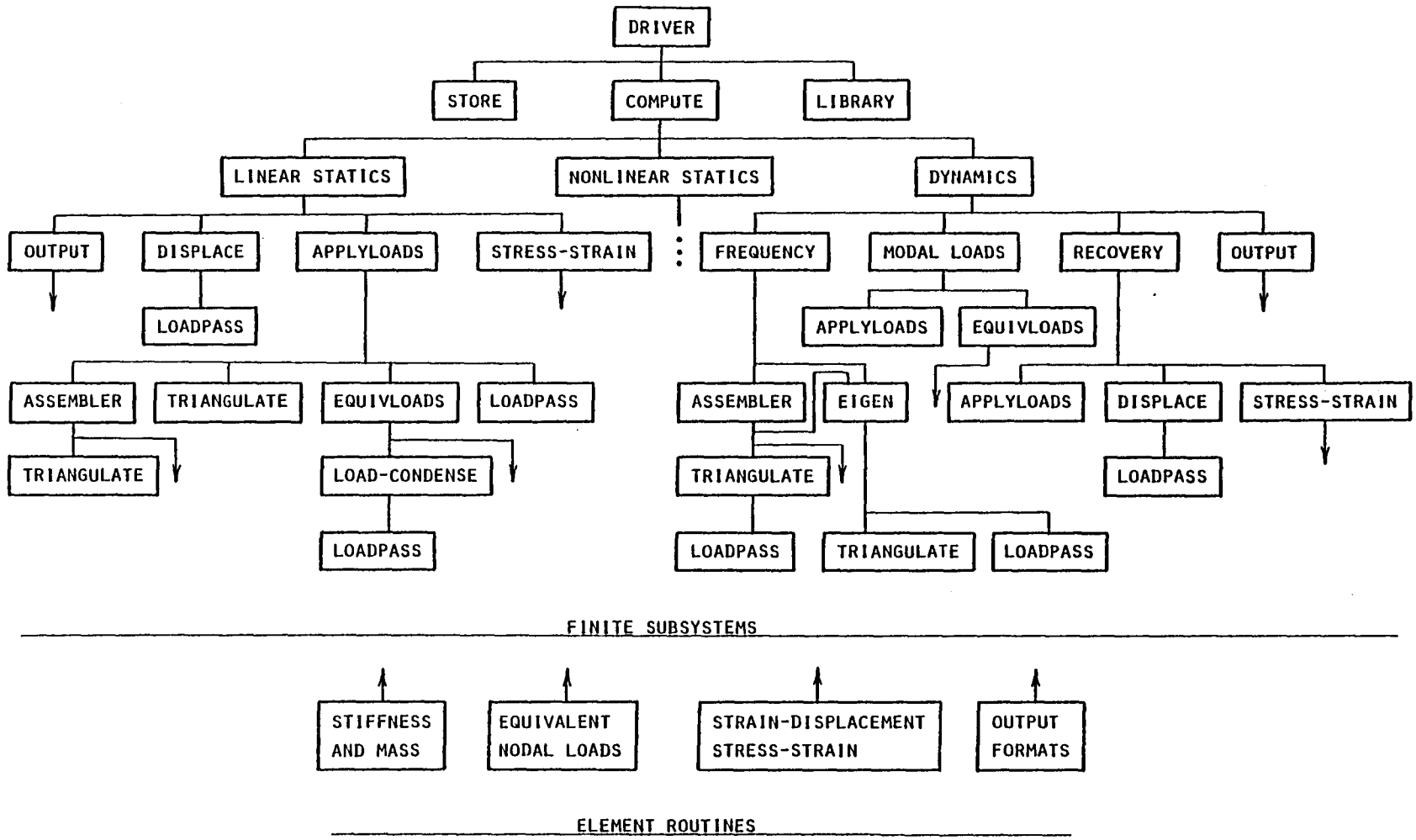


Figure 4.2. Functional Dependencies Among the FINITE Sybsystems

Requests for computation and output are passed to the COMPUTE subsystem. COMPUTE determines which type of analysis is required and invokes the appropriate processors (LINEAR STATICS, NONLINEAR STATICS, and LINEAR DYNAMICS). For brevity, the NONLINEAR STATICS processor and the nonlinear material models are not presented in detail (see [16] for a detailed description). The LINEAR STATICS branch is illustrated in Figure 4.2 and the DYNAMICS branch is fully described in Section 4.5.

The subprograms which perform the computations for the various finite elements are linked to the FINITE subsystems independently of POLO. Subsystems such as ASSEMBLER and STRESS-STRAIN invoke the element routines directly through FORTRAN subroutine calls. The element routines are written entirely in standard FORTRAN. For instance, all element stiffness routines have the same set of formal parameters. This approach allows developers to implement new elements without detailed knowledge of the FINITE system structure. Only the standardized form of the subroutine call is needed to link a new element into the system.

4.2.2 Application Databases

The data structures for FINITE are logically partitioned into three functional units named: LIBRARY, STRUCTURE, and SOLVER (Figure 4.1). Each of these units is defined via the POLO DDL. During execution of FINITE to analyze a structure, a direct access file is "formatted" with each DDL to initialize the databases.

The LIBRARY database contains the tables that describe the element definitions and material models which have been implemented by the system developers. Other data on this database include geometric

properties of steel sections. During solution of a typical finite element problem, this database is open in "read-only" mode.

The STRUCTURE database contains the internal form of all user supplied structure information, along with computed results such as stiffness, mass, loads, displacements, frequencies and mode shapes. An extensive hierarchical data structure, similar to that shown in Figure 3.2, is defined for the logical organization of this data.

The SOLVER database contains data for numerical computations such as triangulations, load-passing, and eigenproblem solution. Information in the STRUCTURE database is moved to the SOLVER and reformatted in hypermatrix form to support efficient numerical computation. After a request for computation has been satisfied, intermediate results, such as the triangulated stiffness, are retained in the SOLVER database for possible use in analysis restart. Other results, such as mode shapes, are transfered into tables associated with each structure node and copied back to the STRUCTURE database. Section 4.4 describes hypermatrix data structures and provides examples of SOLVER data tables used in frequency analysis.

The separation of problem data into two databases, STRUCTURE and SOLVER, is an arbitrary choice. It was done to anticipate size limitations on direct access files in some computer systems and to increase flexibility in placement of the data on peripheral devices.

4.2.3 Subsystem Interfacing

As shown in Figure 4.2, one subsystem can be initiated by several other subsystems. For example, subsystem TRIANGULATE can be initiated by APPLYLOADS, ASSEMBLER, and EIGEN. The point at which TRIANGULATE is

initiated is determined as the analysis of a structure progresses. The key issue is that the individual subsystems are self-contained, which makes this flexibility possible. POLO provides only limited support for communication between subsystems. Only information contained in the databases is handled automatically. No facility exists that parallels the argument list of FORTRAN subroutine calls.

To overcome this deficiency, a "request" vector scheme has been implemented throughout the FINITE subsystems. The request vector is generated by the calling subsystem and is placed at the top of a request "stack." The stack resides in the STRUCTURE database and is thus accessible to all FINITE subsystems. After generation of the request vector, execution of the calling subsystem is interrupted and the new subsystem is initiated. The new subsystem takes its instructions from the request vector at the top of the stack. The request vector contains data such as the name of the current structure, the loading condition name, and the type of request (compute displacements, output stresses, etc.).

When a subsystem completes execution, control must be returned to the subsystem that made the call. This function is supported by the POLO executive. POLO maintains its own stack of executed subsystems so the order of execution can be re-traced. This process is identical to the management of multiple levels of subroutine calls in standard FORTRAN programs.

4.3 User Interface for Dynamic Analysis

Implementation of the dynamic analysis capabilities required extension of the FINITE user interface. Several of these extensions are illustrated by use of the bridge example from Chapter 1. Figures 1.2

and 1.4 (repeated in Figures 4.3 and 4.4) illustrate the structural model and present input to define the model for analysis. With the application of structure loads, a static analysis could be performed to compute nodal displacements and element strains and stresses. Additional structure characteristics and analysis parameters are needed for dynamic analysis. The following is a discussion of specific input commands for frequency and mode shape computation. Full details of the input commands for dynamic analysis are given in Appendix A.

The first addition to the model definition is the specification of the mass of each element and structure in the hierarchy. The mass of a structure is considered in two parts: primary and secondary. Primary mass is the mass of the load-carrying components (elements) of the structure. Primary mass is defined in the POL through definition of a mass formulation indicator: LUMPED or CONSISTENT, and a new element property: MASS_DENSITY. The element definition command for the simple elements in structure SPAN becomes:

```
ELEMENTS ALL TYPE ROD  LUMPED  E 3.0E04  AX 0.0347,
                       MASS_DENSITY 7.34E-04
```

A similar command is used for elements 3-5 in structure BRIDGE. Definition of primary mass is necessary only for finite elements. The primary mass for a structure is assembled from that of the elements which form the structure. Assembly of a structure's primary mass follows a procedure identical to that used in stiffness assembly. Structures which are composed of condensed lower level substructures obtain their mass definition directly through the condensation process. The FINITE system accepts up to thirty DOF at each node in the structure. These are the displacement DOF (u v and w) plus their first

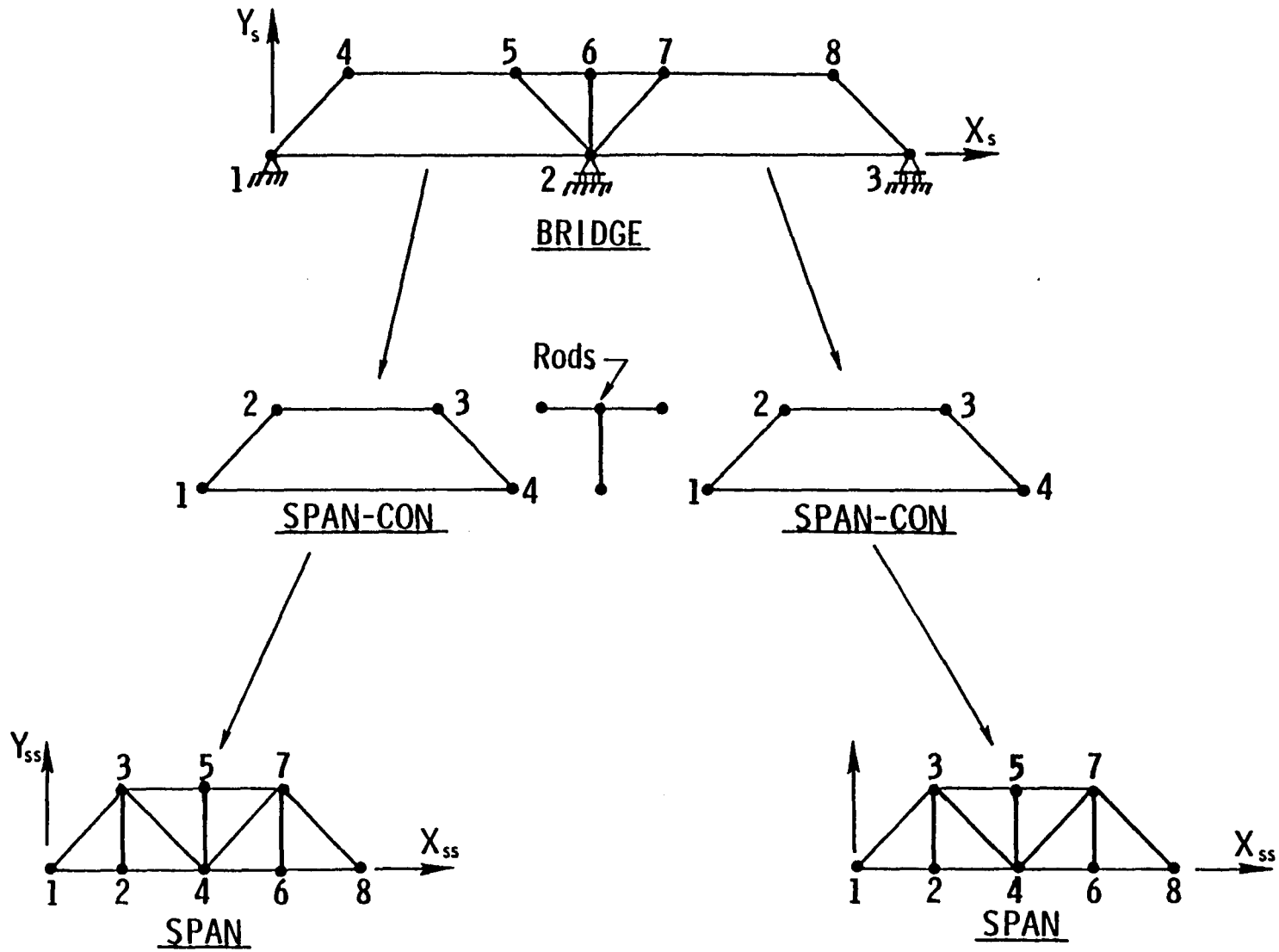


Figure 4.3. Substructured BRIDGE Model

```
*RUN FINITE
C
C          DEFINE THE BRIDGE SEGMENT:  SPAN.
C          UNITS ARE "KIPS" AND "FEET".
C
  STRUCTURE SPAN
    NUMBER OF NODES 8 ELEMENTS 13
    ELEMENTS ALL TYPE ROD  E 3.0E04  AX 0.0347
    COORDINATES
      1     0.0   0.0
      2    20.0   0.0
      3    20.0  20.0
      4    40.0   0.0
      5    40.0  20.0
      6    60.0   0.0
      7    60.0  20.0
      8    80.0   0.0
    INCIDENCES
      1   1   3
      2   2   3
      3   3   4
      4   4   5
      5   4   7
      6   6   7
      7   7   8
      8   3   5
      9   5   7
     10   1   2
     11   2   4
     12   4   6
     13   6   8
C
C              END OF STRUCTURE SPAN
C
C
C              DEFINE THE CONDENSED VERSION OF STRUCTURE SPAN.
C              RETAIN NODES 1 3 7 AND 8 IN THE CONDENSED STRUCTURE.
C
  STRUCTURE SPAN_CON
    NUMBER OF NODES 4 ELEMENTS 1
    ELEMENT 1 TYPE SPAN CONDENSED
    INCIDENCES
      1   1  3  7  8
C
C              END OF STRUCTURE SPAN_CON
C
C
C          DEFINE THE HIGHEST LEVEL STRUCTURE AS A COMBINATION
C          OF TWO CONDENSED SPANS AND THREE SIMPLE ROD ELEMENTS
C
C
  STRUCTURE BRIDGE
C
    NUMBER OF NODES 8  ELEMENTS 5
    ELEMENTS
      1 2  TYPE SPAN_CON ROTATION SUPPRESSED
      3-5  TYPE ROD E 3.0E04  AX 0.0347
C
    COORDINATES
      2     0.0    0.0
      5   -20.0   20.0
      6     0.0   20.0
      7    20.0   20.0
C
    INCIDENCES
      1     1  4  5  2
      2     2  7  8  3
      3     5  6
      4     6  7
      5     2  6
C
    CONSTRAINTS
      1-3  V  =  0.0
      1    U  =  0.0
C
    <definition of loads>
C
    <requests for computation>
C
    <requests for output>
C
    STOP
```

Figure 4.4. POL Definition of BRIDGE Model

and second spatial derivatives ($u_{,x}$ $v_{,x}$ $w_{,x}$ $u_{,xx}$ ...). Depending upon the type of elements incident on a given structure node, it is possible for mass to be assigned to any or all of these DOF.

Secondary mass is the mass of non-load-carrying objects supported by the structure. Examples include water in a tank and mechanical equipment in a building. Secondary mass is always treated as a lumped mass addition to the primary mass of the structure. There are two types of secondary mass: nodal mass and element mass. Nodal mass is concentrated at a structure node. Element mass is concentrated or distributed on the surface of an element. Element mass is resolved into equivalent nodal mass by use of the same shape functions that resolve element loads into equivalent nodal loads. As with primary mass, secondary mass may be assigned to any of the applicable nodal DOF. Application of secondary mass to structure SPAN could take the form:

```
MASS
     NODAL
          2 4 6  U V  1.23
     ELEMENT MASS FOR ELEMENT TYPE PLANEFRAME
          1-3 LINEAR U V W FRACTIONAL LA 0.0 LB 1.0 WA 0.0 WB 0.5
```

By this command sequence, nodes 2, 4, and 6 have mass of 1.23 units applied to the U and V (translational) DOF. Also, elements 1-3 have a linearly varying mass distributed along their length. The mass intensity is 0.0 at the beginning of the elements and increases to 0.5 at the end. The secondary mass command sequence is grouped with the definition of COORDINATES, INCIDENCES, CONSTRAINTS, and LOADS.

Before frequency analysis of a structure can be performed, an analysis method must be selected. In general, no single method is appropriate for all structures in a complex hierarchy. Since eigenproblem

solution is normally a computationally expensive procedure, it is wise to select an analysis method that is well suited to the structure being analyzed. An analysis procedure that is effective for a small model with a fully populated stiffness matrix will not generally be efficient in the analysis of a large model with a tightly banded stiffness. Since this broad variety of structures may exist within one structural hierarchy, the analyst must have the capability to define a unique analysis procedure for each structure for which frequency analysis will be performed. Such a capability has been implemented in FINITE. Specification of the analysis method for structure SPAN may take the form:

```
FREQUENCY ANALYSIS TYPE JACOBI
```

where the generalized Jacobi method [4] is selected and default values for convergence tolerance and maximum number of sweeps are implied. As a second example, structure BRIDGE may require the following analysis definition:

```
FREQUENCY ANALYSIS TYPE SUBSPACE
     PROPERTIES    NUMBER OF PAIRS 4  ITERATIONS 10,
                   SUBSPACE SIZE 8    STURM CHECK
```

In this command sequence, the subspace iteration method [58] is selected and the default values are used for all properties not specified. These sample commands are used to define the frequency analysis method and the associated parameters that control the solution. The frequency analysis is invoked by one of two procedures. First, the analyst may enter an explicit "COMPUTE FREQUENCIES" or "COMPUTE MODE SHAPES" request. Second, a frequency analysis can be invoked automatically within FINITE to satisfy a computational request involving a substructured model. For example, condensation of structure SPAN to produce structure SPAN_CON

may require fixed-fixed mode shapes and frequencies for structure SPAN. In this case, the appropriate FINITE processor invokes the frequency analysis of SPAN simply as another step in the condensation process.

As discussed in Chapter 2, condensation by the fixed-interface method requires retention of a substructure's master DOF plus a selected number of generalized (normal) DOF. For static analysis, the definition of structure SPAN_CON uses an incidence list to identify the master DOF from structure SPAN that are retained during static condensation (see Figure 4.4). This same technique is used in defining master DOF in structures to be reduced by the fixed-interface method. Substructure normal DOF are defined by expanding the element definition command as follows:

```
ELEMENT 1 TYPE SPAN  CONDENSED  RETAIN NORMAL MODES 1-3
```

In this example the lowest three frequencies and mode shapes computed in a fixed-fixed analysis of SPAN are used to compute the three generalized coordinates retained in SPAN_CON. This format for definition of condensed substructures is used at all levels of the structure hierarchy.

The complete format for the above commands plus the associated computation and output requests are presented in Appendix A. The commands for analysis capabilities which have not yet been implemented (transient analysis, shock spectrum response, non-zero initial conditions, etc.) are also presented to illustrate the ease with which these functions can be defined by the analyst. Implementation of these additional capabilities remains a topic for future study.

4.4 Data Structures for Dynamic Analysis

A variety of new data structures was designed for the implementation of dynamic analysis in FINITE. The following sample data structures are presented to illustrate their effects on the numerical algorithms.

The sample data structure in Figure 3.2 (repeated in Figure 4.5) is typical of that used to store user input and computed results in the STRUCTURE database. An extensive hierarchy of pointers is established to isolate numerical data in relatively small quantities. For example, stiffness and mass matrix coefficients for both elements and structures are grouped on a node by node basis. An advantage to this approach is that element matrices and substructure matrices have identical formats. Therefore when stiffness and mass matrix assembly is performed, the same procedure is used for both finite elements and substructures. This feature has a major influence on the implementation of multilevel substructured modeling. The more complex data structures permit the development of simpler, yet more general, computational processes.

A disadvantage of this data structure is the overhead incurred by the data manager in accessing the small matrices many times in triangulation, loadpass, and eigensolution. Since the data are stored in small blocks, the data manager is executed more frequently than if the data were in larger blocks. To alleviate this problem, required data are moved to the SOLVER database and reformatted into hypermatrices prior to performing numerical computations. Thus, the use of the SOLVER database permits optimal allocation of the equations on secondary storage to minimize the time spent in data management during the solution phase.

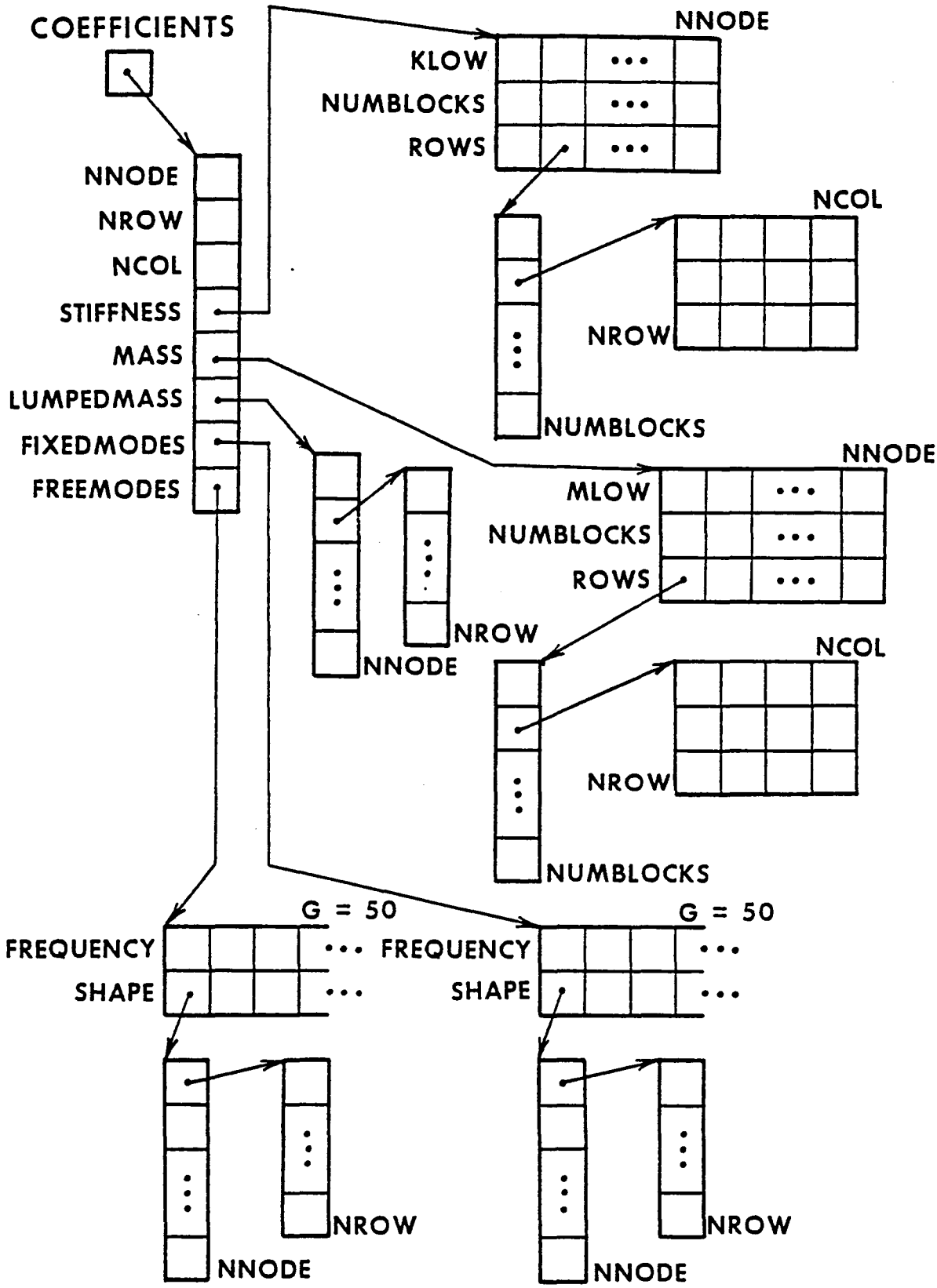


Figure 4.5. Sample Data Structure

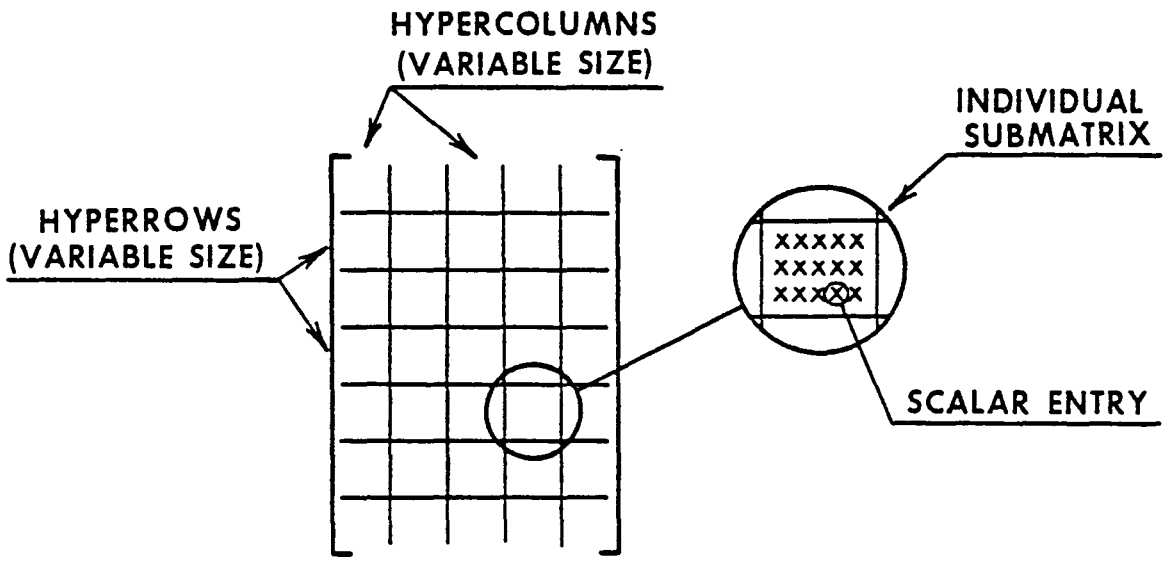
4.4.1 Hypermatrix Data Structures

Hypermatrices provide the fundamental data structure used in FINITE to support equation solving (triangulation and load-pass) and eigensolution. A matrix which is partitioned by rows and columns into submatrices is called a hypermatrix. Figure 4.6 illustrates hypermatrix partitioning and the corresponding data structure for storing and retrieving the individual submatrices. The order of each submatrix is determined by the number of rows assigned to each hyperrow and the number of columns assigned to each hypercolumn. These assigned values are selected to produce a balance among the overhead in accessing the submatrices, I/O performance, and memory requirements. The potential for zero entries in a submatrix from a banded hypermatrix also influences the size of the partitions. In general, the order of each submatrix may vary from hyperrow to hyperrow and from hypercolumn to hypercolumn. Currently, the maximum sizes of an individual submatrix in FINITE are 60 rows and 60 columns.

The data structure adopted to represent a hypermatrix is shown in Figure 4.6b. The first-level pointer vector contains row pointers, each of which locates data in the corresponding hyperrow. The second-level vector of pointers, the column pointers, identifies the location of each submatrix on the hyperrow. Two sizing vectors are used to store the number of rows in each hyperrow and the number of columns in each hypercolumn.

Banded, symmetric hypermatrices (such as the structure stiffness and mass) are partitioned as illustrated in Figure 4.7a. Only submatrices in the lower triangle of the matrix are stored. Zero

A. PARTITIONING OF A HYPERMATRIX



B. HYPERMATRIX DATA STRUCTURE

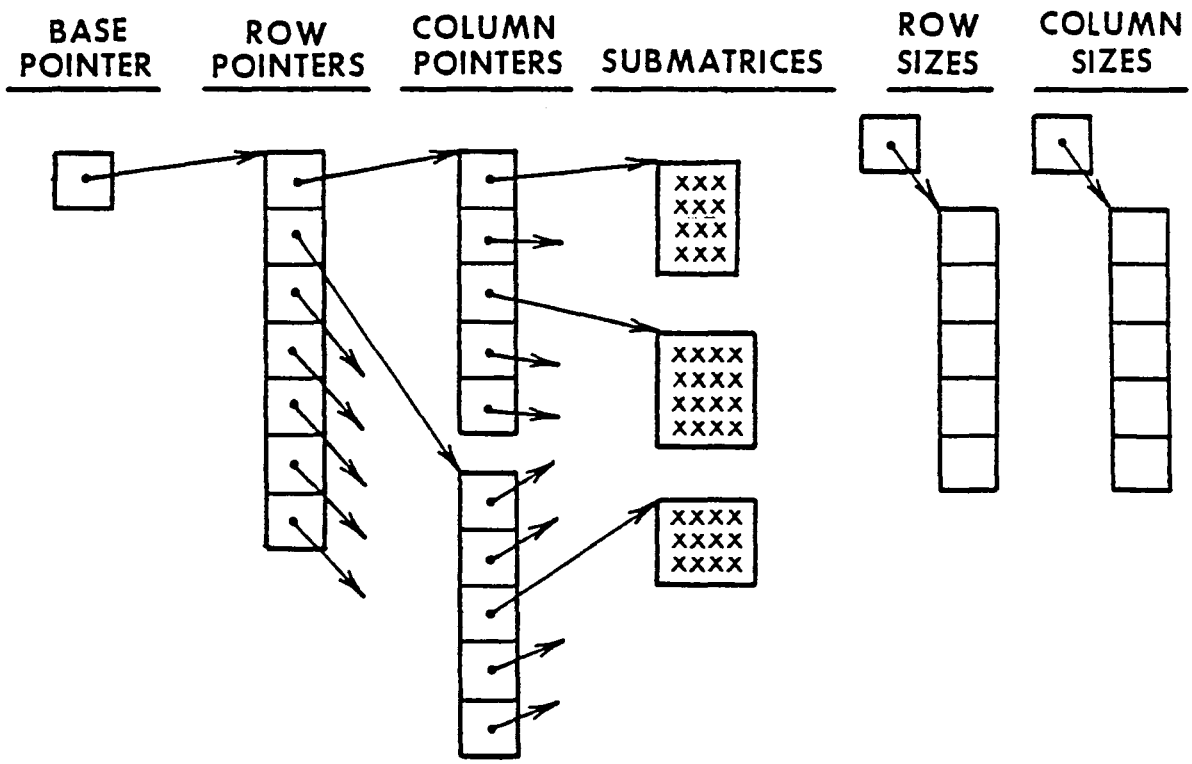
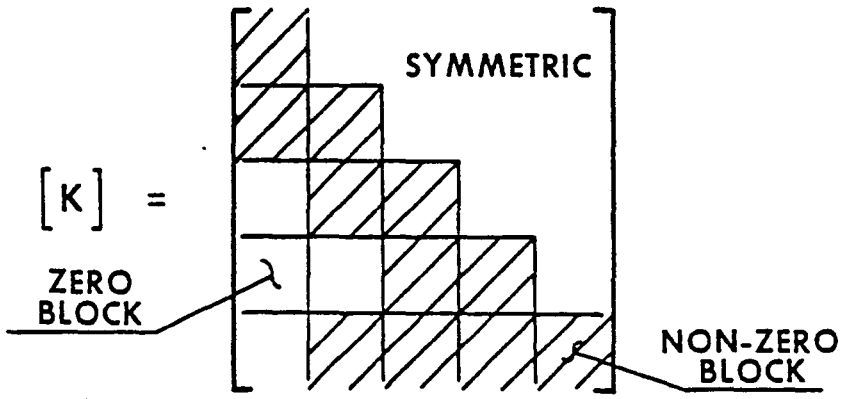


Figure 4.6. Representation of a Hypermatrix

A. PARTITIONED STIFFNESS MATRIX



B. ASSOCIATED DATA STRUCTURE

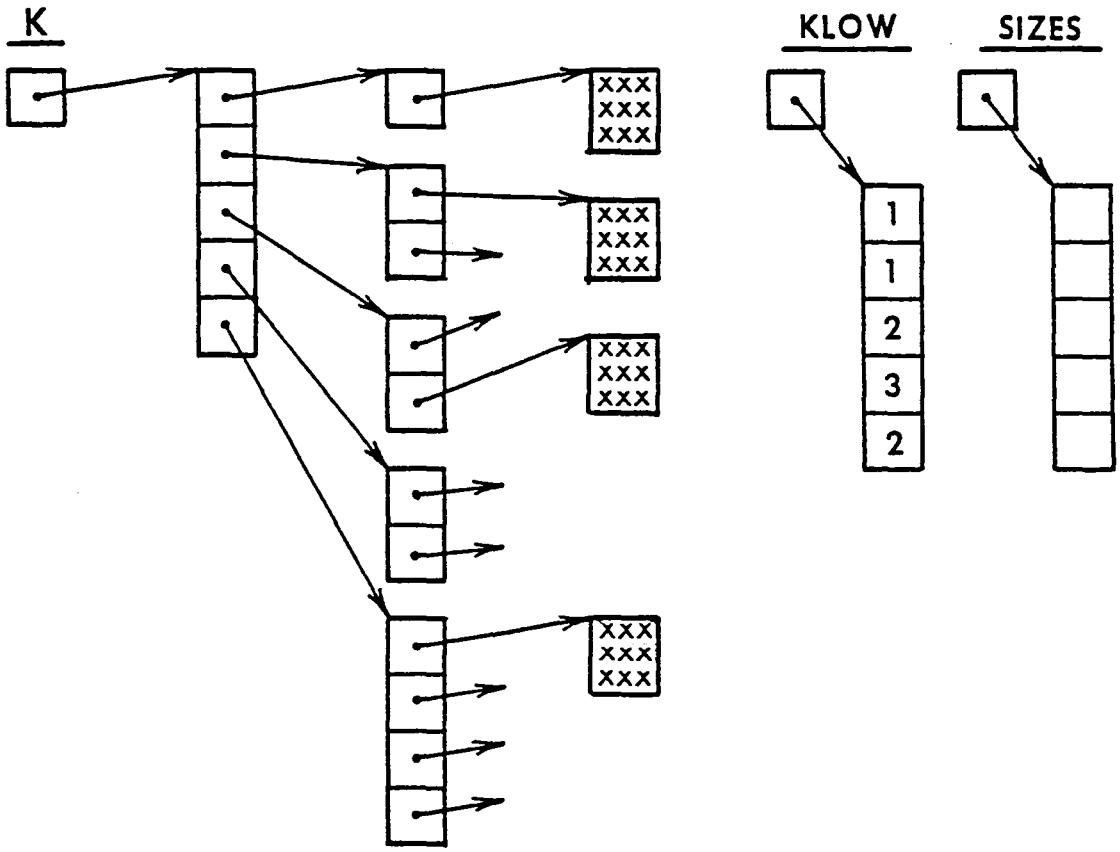


Figure 4.7. Banded, Symmetric Hypermatrices

submatrices outside the band of the matrix are not created. Zero submatrices within the band are created since the submatrices become non-zero during computations. When a symmetric matrix is partitioned, hyperrow and hypercolumn sizes are selected so that the diagonal submatrices are square, thus yielding a symmetric partition.

The data structure for banded, symmetric hypermatrices is similar to that for general hypermatrices. As shown in Figure 4.7b, a two-level pointer hierarchy is used in which the first-level pointer vector locates data on the hyperrow. For banded, symmetric hypermatrices, the column pointers locate data from the first non-zero submatrix on the hyperrow through the diagonal submatrix. Since the symmetric partition produces row-sizes and column-sizes vectors that are identical, a single sizing vector is sufficient. The banding information is contained in a vector called KLOW. KLOW contains one integer entry for each hyperrow in the hypermatrix. This integer defines the hypercolumn subscript for the first non-zero submatrix on the hyperrow. Using Figure 4.7 as an example, the first non-zero submatrix on hyperrow 4 occurs in hypercolumn 3. Thus the fourth entry in vector KLOW is 3.

The data structure described above is just one of several ways to represent a hypermatrix in a hierarchical form. One alternative is presented in [19] in which the submatrix pointers are stored in a pointer matrix rather than in a two-level pointer hierarchy. This technique allows the pointer matrix itself to partitioned into a hypermatrix creating a multilevel hypermatrix data structure. While an exhaustive study has not been made to identify the optimum technique (if one does indeed exist), the foregoing data structure has proven to be effective in FINITE. Even though hypermatrix data structures minimize data

management overhead, the total number of data words transfered between memory and secondary storage may actually increase. This is because the blocking procedures require the addition of extraneous zero terms to the database. Remedies to this problem are discussed in the following section.

4.4.2 Hypermatrix Solution Algorithms

An advantage of hypermatrix data structures is that submatrices of a hypermatrix can be accessed as efficiently column-wise as row-wise. In contrast to column (or skyline) storage of sparse matrices, a hypermatrix can be used effectively as a pre-multiplier, as a post-multiplier, and as its own transpose [19]. In a virtual memory environment, no paging penalties are incurred when performing matrix multiplication, triangulation, and load-pass operations so long as no more than one submatrix occupies a physical record (page) on secondary storage.

Computations on hypermatrices typically require no more numerical operations than the same computations on conventionally stored matrices. Economical solutions can be achieved when proper account is made of operations on zero entries in the non-zero submatrices and when data accessing procedures are tailored to the specific application. As an example, consider the triple-matrix product performed in subspace iteration. The transformation of the mass matrix from geometric coordinates to subspace coordinates is

$$[\bar{M}] = [\bar{X}]^T[M][\bar{X}] \tag{4.1}$$

where [M] is the structure mass matrix, $[\bar{X}]$ is the set of iteration vectors, and $[\bar{M}]$ is the transformed mass. The conventional approach to this transformation is to compute the product

$$[T] = [M][\bar{X}] \tag{4.2}$$

followed by the product

$$[\bar{M}] = [\bar{X}]^T[T]. \tag{4.3}$$

With this approach, the intermediate product [T] must be computed and held in memory or on secondary storage until all computations are complete.

An alternative approach to implementation of the triple-matrix product does not require the temporary matrix [T]. Assume that [M] is partitioned as a hypermatrix with "n" hyperrows and "n" hypercolumns and that $[\bar{X}]$ is partitioned into "n" hyperrows and "q" hypercolumns (q<<n for most applications). The following algorithm requires only a temporary submatrix [S] to perform the triple product.

```
DO i = 1, n
  DO k = 1, q
    [S] = [0]
    DO j = 1, n
      [S] = [S] + [M_ij][X̄_jk]
    END DO
    DO j = 1, n
      [M̄_jk] = [M̄_jk] + [X̄_ij]^T[S]
    END DO
  END DO
END DO
```

In the above, the subscripts identify the hyperrow and hypercolumn from which the associated submatrix is taken. This algorithm builds the

product $[\bar{M}]$ incrementally where the temporary product in [S] is used as soon as it is computed.

The algorithm is modified to recognize leading zeros in the mass submatrices as follows. When the submatrix product $[M_{ij}][\bar{X}_{jk}]$ is computed, $[M_{ij}]$ is examined to locate the first non-zero entry on each row. The corresponding column subscript is then used as a lower bound for the inner loop of multiplies to avoid operations on zero entries. Since the matrix of iteration vectors, $[\bar{X}]$, is fully populated, no tests are performed on the entries in $[\bar{X}_{ij}]$ prior to computation of $[\bar{X}_{ij}]^T[S]$.

The above algorithm is for the case when [M] is fully populated and all submatrices are stored (lower and upper triangle). When [M] is stored as a banded, symmetric hypermatrix, subscript adjustments are necessary to properly access the $[M_{ij}]$ submatrices.

There is no significant difference in operation counts between this algorithm and the procedure of equations (4.2) and (4.3). Also, the number of submatrices accessed is the same for each algorithm. The advantage of the new algorithm is that memory and secondary storage requirements are minimized by eliminating the need for the temporary hypermatrix [T]. The above procedure provides another advantage when implemented on computers with virtual memory. The use of hypermatrices serves to minimize operating system paging. Since the submatrices are of moderate size, all entries in the submatrix can normally be accessed without the need for paging by the operating system. Conventional matrix products require row-wise data access and thrashing may result when the matrices are large.

4.5 Subsystem DYNAMICS

Several new subsystems were needed for the implementation of dynamic analysis capabilities in FINITE. Likewise, most of the existing subsystems required either minor or major modification to handle the new data structures and computational procedures. For example, subsystem OUTPUT was simply extended to support output of natural frequencies, mode shapes, modal loads, and modal strains and stresses. In contrast, subsystem ASSEMBLER required major revision to combine mass matrix assembly with stiffness assembly and to include the use of normal DOF in both matrices. As mentioned earlier, it is impractical to review all the details of the implementation. Instead, the remainder of this chapter presents a selection of the software developed for the study. Both new subsystems (DYNAMICS and EIGEN) and modifications to existing subsystems (ASSEMBLER, TRIANGULATE, and LOADPASS) are discussed.

In dynamic analysis, requests for computation and output are passed to subsystem DYNAMICS by subsystem COMPUTE (see Section 4.2.1 and Figure 4.2). Subsystem DYNAMICS controls the processors that provide the dynamic analysis capabilities of FINITE. When a "dynamics" request is received, the request vector is examined to determine which function is requested and which structural hierarchy is involved. DYNAMICS then invokes lower level subsystems to satisfy the request. Current capabilities of subsystem DYNAMICS include frequency analysis, computation of modal loads, recovery of computed results for condensed substructures, and output of the various computed results. These capabilities are managed by four separate subsystems, as shown in Figure 4.2. They are FREQUENCY, MODAL_LOADS, RECOVERY, and OUTPUT. The following is a brief overview of the first three of these subsystems.

Subsystem OUTPUT required only simple extension to support the various dynamics output requests, so it is not described here.

Frequency analysis entails the computation of natural frequencies and mode shapes for a structure at any level of the structural hierarchy. Frequency analysis is preceded by assembly of the stiffness and mass matrices for the structural model. For a standard (non-substructured) model, assembly is performed without interruption and the frequency analysis (subsystem EIGEN) is then invoked. The logical flow through the subsystem hierarchy in Figure 4.2 is the following. When a request for frequency analysis is translated by subsystem DRIVER, control is transfered from DRIVER to COMPUTE to DYNAMICS to FREQUENCY. Subsystem FREQUENCY invokes ASSEMBLER to perform the stiffness and mass assembly. Since the model does not include substructures, subsystem ASSEMBLER performs the assembly without invoking any other subsystems (only element stiffness and mass routines are called). When ASSEMBLER terminates, control is returned to FREQUENCY. FREQUENCY then invokes subsystem EIGEN to perform the frequency analysis. When EIGEN terminates, control is transfered back to FREQUENCY, which in turn returns control to DYNAMICS and so on.

If the structural model contains condensed, lower level substructures, the condensation and assembly procedure requires ASSEMBLER to run other subsystems. For fixed-interface reduction of a substructure, subsystem ASSEMBLER interrupts its own execution and invokes subsystem EIGEN to perform the fixed-fixed frequency analysis of the substructure. When EIGEN terminates, control is returned back to ASSEMBLER. Subsystem TRIANGULATE is then initiated to perform the reduction. After the stiffness and mass matrices for the current substructure have been reduced,

control is again returned to ASSEMBLER and the assembly process continues. This process is performed recursively until all structures in the hierarchy have been condensed and assembled. When the entire structural hierarchy has been assembled, ASSEMBLER terminates and control is returned to FREQUENCY. At that point, subsystem EIGEN is again invoked to solve the eigenproblem for the highest level structure. Details of the frequency analysis and condensation procedures follow later in this chapter.

Computation of modal loads requires simply a transformation of a load vector (in geometric coordinates) to modal coordinates. The load vector is obtained from the prior definition of a loading condition by the analyst. The mode shapes computed in a frequency analysis of the structure are used for the transformation from geometric to modal coordinates. The modal loads processor permits the analyst to identify those vibration modes that are most likely to participate in the response of the structure under a given dynamic load. This information is useful in performing transient analysis by mode superposition. Full implementation of mode superposition capabilities is not included in this study.

After frequencies and mode shapes have been computed for the highest level structure in a substructured model, the mode shapes for condensed lower level substructures may be recovered. The necessary procedures are managed by subsystem RECOVERY. A request for computation or output of mode shapes, modal strains, or modal stresses causes RECOVERY to be invoked. The transformation matrix of static constraint modes and substructure normal modes is used to transform the mode shapes from the reduced set of generalized coordinates back to the geometric

coordinates of the uncondensed substructure (see Equation 2.7). This process is repeated recursively until the lowest level of the hierarchy is reached. At this point, the portion of the mode shape which corresponds to the condensed substructure DOF can be output to the analyst.

Recovery of modal strains and modal stresses is performed after mode shape recovery. Modal strains are the strains computed for the individual finite elements when a free-vibration mode shape is used as a displacement vector. Modal stresses are derived from modal strains through the stress-strain relations for the element. Computation of modal strains is useful in evaluation of the modeling and analysis procedures, as is discussed in the next chapter.

4.6 Frequency Analysis

The efficiency and flexibility of the dynamics capabilities of FINITE depend heavily upon the capabilities of the eigenproblem solver. For this reason, frequency analysis is discussed in more detail than the previous topics.

Computation of natural frequencies and mode shapes has been implemented in FINITE in the form of two eigenproblem solvers: the generalized Jacobi method and subspace iteration. Computations for both eigensolvers are managed by subsystem EIGEN. EIGEN may be invoked to solve the eigenproblem for structures at any level of the structural hierarchy and with any specified boundary conditions. This includes fixed-fixed frequency analysis for condensed substructures and free-vibration analysis for constrained or unconstrained structures. Subsystem EIGEN determines the nature of the analysis from the characteristics of the structure and from instructions contained in the

request vector. The particular solution method which is used (JACOBI or SUBSPACE) is selected by the analyst. Each of the two eigensolvers is discussed below. Data structures and details of the algorithms are described.

4.6.1 Generalized Jacobi Method

The computation of natural frequencies and mode shapes for discrete structural models is achieved by solution of the generalized eigenproblem:

$$[K][\varphi] = [\omega^2][M][\varphi] \tag{4.4}$$

where [K] and [M] are symmetric, positive definite coefficient matrices, $[\varphi]$ is the matrix of eigenvectors, and $[\omega^2]$ is the diagonal matrix of eigenvalues. The generalized Jacobi method [4] is one of two eigensolvers implemented in FINITE for solution of this problem. The generalized Jacobi method serves two functions in FINITE. First, it is used to compute all frequencies and mode shapes for small structural models. Second, the method is used as a component of subspace iteration. The generalized Jacobi method is popular because of its simplicity and its ability to handle ill-conditioned or singular coefficient matrices.

In the generalized Jacobi method, [K] and [M] are iteratively transformed using orthogonal rotation matrices to zero the off-diagonal terms in each matrix. After sufficient iteration, the matrices are driven to diagonal form and the eigensolution is complete, yielding all eigenpairs for the problem. Convergence of the method is quadratic once the off-diagonal elements are small. Thus a high degree of accuracy in

the solution can be achieved by continued computation at little additional cost. This characteristic has made the generalized Jacobi method an efficient component of the subspace iteration method (discussed in the next section).

Implementation of the generalized Jacobi method in FINITE required a limitation on the basic formulation presented in [4]. The order of the problem which can be solved is currently limited to 60 DOF. This restriction assures that the stiffness and mass matrices will each occupy only one submatrix. This yields a memory-resident solution procedure. Since the generalized Jacobi method loses efficiency when the order of the problem is large, a corresponding hypermatrix formulation which requires additional I/O is of questionable value [7].

4.6.2 Conventional Subspace Iteration

The subspace iteration method [3] is used to compute the "p" lowest eigenvalues and corresponding eigenvectors for the generalized eigenproblem, Equation (4.4). In this case, [K] and [M] have order nxn, $[\varphi]$ has order nxq, and $[\omega^2]$ has order qxq (q>p). The method belongs to the simultaneous iteration class of eigenproblem solvers in which inverse iteration is performed with a set of orthogonal iteration vectors. In subspace iteration, a special Ritz analysis is performed to enforce orthogonality of the iteration vectors and to enhance convergence.

The first step of the method is to select a set of "q" iteration vectors that reside in the nxq matrix [X]. When the method was initially proposed, "q" was selected as the minimum of "2p" and "p+8." Using "q" iteration vectors instead of just "p" vectors improves the

convergence rate for the p^{th} eigenvalue from $(\omega_p^2/\omega_{p+1}^2)$ to $(\omega_p^2/\omega_{q+1}^2)$. Next, [K] is triangulated such that

$$[K] = [L][L]^T \tag{4.5}$$

where [L] is the lower triangular Choleski factor of [K]. After triangulation, the iteration cycle begins.

Compute the inertia-load vectors

$$[F] = [M][X]. \tag{4.6}$$

Find the pseudo-displacements corresponding to the inertia loads by solving for $[\bar{X}]$ in

$$[L][L]^T[\bar{X}] = [F]. \tag{4.7}$$

Transform the stiffness and mass to subspace coordinates by

$$[\bar{K}] = [\bar{X}]^T[F] \quad \text{and} \tag{4.8}$$

$$[\bar{M}] = [\bar{X}]^T[M][\bar{X}]. \tag{4.9}$$

Using the generalized Jacobi method, solve the qxq eigenproblem for the subspace

$$[\bar{K}][\Psi] = [\lambda][\bar{M}][\Psi]. \tag{4.10}$$

Finally, compute the improved iteration vectors [X] as

$$[X] = [\bar{X}][\Psi]. \tag{4.11}$$

The result of equation (4.11) is used in equation (4.6) to start the next iteration. Convergence is achieved when the first "p" eigenvalues in $[\lambda]$ do not change (by more than a tolerance) from one iteration to the next. At convergence, $\lambda_i \rightarrow \omega_i^2$ and $\{X_i\} \rightarrow \{\varphi_i\}$ for $i = 1, \ldots p$.

Equations (4.6) and (4.7) form the simultaneous inverse iteration steps, while equations (4.8) - (4.11) define the Ritz analysis.

Selection of the initial iteration vectors may be based on a number of different procedures. The simplest approach is the following. Entries in the first column of [X] are taken as the diagonal terms of [M]. The remaining columns of [X] are unit vectors with 1.0 entries at coordinates with the largest m_{ii}/k_{ii} ratios. This procedure attempts to excite the modes with the lowest natural frequencies.

The conventional subspace iteration method was not developed in conjunction with any particular data structure. During implementation the numerical procedure must be modified to be compatible with the chosen data structures. A modified subspace iteration procedure was developed, based on the work of other researchers, to conform to hypermatrix data structures.

4.6.3 Hypermatrix Subspace Iteration

In spite of its popularity, several problems have been identified with the use of the conventional subspace iteration method [58]. The most significant of these is the computational expense required to form and solve the subspace eigenproblem for large subspace sizes, Equations (4.8 - 4.10). One procedure that has found favor with researchers is the evaluation of eigenpairs in groups with the subspace size, q, less than the number of eigenpairs, p, that are required [5, 35, 58]. The procedure adopted in this study is essentially that presented in [58], in which eigenvectors are removed from the set of iteration vectors as they converge. To keep the subspace size constant, new iteration vectors are introduced to replace the converged vectors. This causes the

domain of the subspace to be shifted to the higher values in the frequency spectrum of the structural model. Therefore, the order of the subspace (q) does not place an upper limit on the number of eigenpairs (p) that may be computed. Origin shifts are also used to improve convergence rates for the higher eigenvalues. The use of hypermatrices in this study has prompted modifications to Wilson's procedure. These modifications are discussed individually, and then the complete hypermatrix formulation is presented.

4.6.3.1 Selection of Iteration Vectors

For the conventional subspace iteration method, initial iteration vectors are selected by identifying the coordinates with the largest m_{ii}/k_{ii} ratios. This approach is not appropriate when the stiffness and mass are stored as hypermatrices. In order to store the ratios, a hypervector data structure is required (see Figure 4.8). Sorting the ratios then requires a multiple-merge sort in which each of the individual subvectors is sorted, then the group of sorted vectors is merged into a single sorted vector. During the entire process, the list of ratios must remain in hypervector form so that it can be transferred to secondary storage as other memory requirements develop.

As an alternative to implementation of the sorting procedure, a new algorithm was developed to select initial iteration vectors. Iteration vectors are chosen as discrete representations of a set of orthogonal cosine functions (see Figure 4.9). This new algorithm guarantees that all unconstrained coordinates will be excited by the inertia loads and that each vector will be orthogonal to the others in the set. This

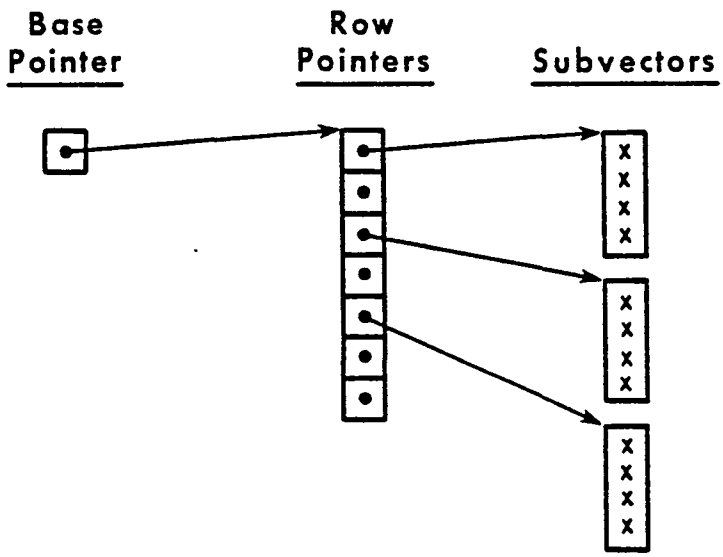


Figure 4.8. Hypervector Data Structure

$$x_{ij} = \cos\left(\pi\left(\frac{i-1}{n-1}\right)\left(j-1\right)\right)$$

i = Row Number
j = Column Number

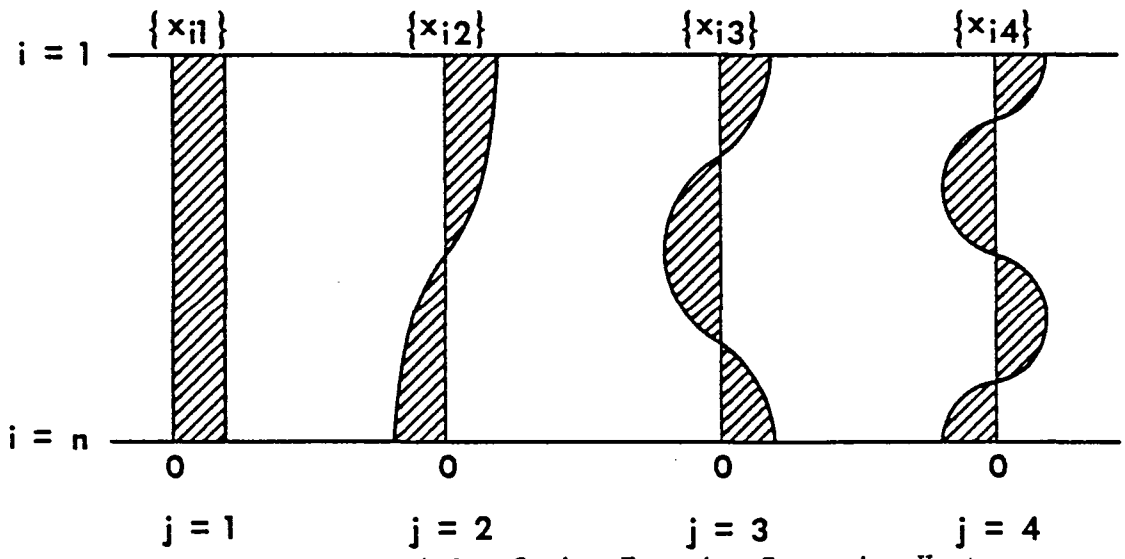


Figure 4.9. Cosine Function Iteration Vectors

procedure is used both for selecting the initial iteration vectors and for generating new iteration vectors to replace converged eigenvectors.

Another procedure for selection of iteration vectors that is compatible with hypermatrix data structures is the use of randomly generated vectors [9]. Although they are simple to generate, the random vectors must be explicitly orghogonalized prior to use in the first iteration.

4.6.3.2 Solution of the Subspace Eigenproblem

The generalized Jacobi method is typically used to solve the eigenproblem for the subspace, Equation (4.10). In conventional subspace iteration, the computational effort required to form and solve the subspace eigenproblem becomes prohibitive as the subspace size increases. Transformation of [K] and [M] to $[\bar{K}]$ and $[\bar{M}]$ requires $(nq^2 + 2nq)$ operations and solution of the subspace eigenproblem requires roughly $(3q^3 + 6q^2)$ operations. Therefore, it is desirable to limit q to maintain efficiency of the overall solution. Yet if q is small, the convergence rate $(\omega_i^2/\omega_{q+1}^2)$ is adversely affected. Selection of q must be based on a balance between a "large" subspace size to obtain good convergence rates and a "small" subspace size to maintain efficiency in the transformations and Jacobi iterations.

Wilson [58] suggested that the optimum subspace size is a function of the bandwidth of the model. This finding provides the basis for a rational approach to the vector replacement procedure reviewed above. To maintain consistency with the generalized Jacobi method and the use

of hypermatrices, the number of iteration vectors (and thus the order of the subspace) is limited to the number of columns that can be placed in one hypercolumn of a hypermatrix. When a set of iteration vectors in [X] is generated, a hypermatrix data structure is used. The hyperrows are sized according to the sizing vector used for [K] and [M] (Figure 4.4), and the number of hypercolumns is limited to just one. When the stiffness and mass transformations are performed using Equations (4.8) and (4.9), the resulting subspace stiffness $[\bar{K}]$ and mass $[\bar{M}]$ each occupy only one submatrix. Thus, the generalized Jacobi procedure can be used as a memory-resident eigensolver for Equation (4.10). Again, the current limit on the order of the subspace eigenproblem is 60x60.

When the stiffness and mass matrices are transformed to subspace coordinates, some terms in $[\bar{K}]$ and $[\bar{M}]$ may become quite large. Additional computations using these terms (such as computing rotation matrix coefficients) may produce exponential overflow. Sources of this problem lie in the units of measure selected by the analyst and in the magnitude of the inertia-load vectors, [F], relative to the structure stiffness [K]. A simple remedy developed in this study involves scaling the subspace stiffness and mass prior to eigensolution. The scale factor is computed as the average of the maximum and minimum exponents of the diagonal terms in $[\bar{K}]$ and $[\bar{M}]$. After eigensolution the scale factor is removed from the eigenvectors $[\Psi]$. The scaling procedure does not affect the eigenvalues $[\lambda]$. Use of this procedure has proven successful in controling exponent growth of the terms in the transformed stiffness and mass.

4.6.3.3 Orthogonalization of Iteration Vectors

When converged eigenvectors are removed from the set of iteration vectors and replacement vectors are inserted, two orthogonalization procedures must be performed. First, the replacement vectors must be mass-orthogonalized to the other iteration vectors in [X]. This operation is performed only at the end of iterations in which replacement vectors are added to [X] due to removal of converged eigenvectors. The purpose of this operation is to force each iteration vector to converge to a different eigenvector. If no convergence occurs during a certain iteration, this orthogonalization step is skipped.

The second orthogonalization procedure guarantees that converged eigenvectors do not reappear in the iteration vectors. Once an eigenvector has been removed from the subspace, all iteration vectors in [X] must be mass-orthogonalized to that eigenvector, and to all other converged eigenvectors. This step must be performed at the start of every iteration following convergence of the first eigenvector.

The Gram-Schmidt procedure is used most often to perform the above orthogonalizations [9, 58]. First, consider orthogonalization of replacement vectors to other iteration vectors in [X]. Assume that two or more replacement vectors have just been added to [X]. The set of vectors can be partitioned to separate "replacement" and "other" vectors:

$$[X] = [X_o \mid X_r]. \tag{4.12}$$

Mass-orthogonalization of $[X_r]$ to $[X_o]$ is achieved by:

$$[\hat{X}_r] = [X_r] - [X_o][X_o]^T[M][X_r]. \tag{4.13}$$

The new set of iteration vectors becomes:

$$[X] = [X_o \mid \hat{X}_r]. \tag{4.14}$$

Notice that the vectors in $[X_r]$ are not mass-orthogonalized to each other as they are added to the subspace. The additional expense of this activity is avoided by selecting replacement vectors which are known to be mutually orthogonal

$$[X_r]^T[X_r] = [I]. \tag{4.15}$$

After orthogonalization with respect to $[X_o]$ by Equation (4.13), the modified replacement vectors $[X_r]$ will converge to the highest eigenvectors within the domain of the subspace:

$$(X_q) \rightarrow (\varphi_s),\ (X_{q-1}) \rightarrow (\varphi_{s-1}),\ \ldots\ , \tag{4.16}$$

where (X_q) is the last vector in $[X_r]$, $s = q+c$, and "c" is the number of converged eigenvectors that have been removed from the subspace so far. Since the convergence rates for these iteration vectors are relatively slow ($\omega_s^2/\omega_{s+1}^2$ for (X_q)), little change in the vectors will occur during the next iteration. At that time, they too will become mass-orthogonal through solution of Equations (4.8) - (4.11).

Mass-orthogonalization of the full set of iteration vectors [X] to the "c" converged eigenvectors in $[\varphi]$ follows the same procedure:

$$[\hat{X}] = [X] - [\varphi][\varphi]^T[M][X]. \tag{4.17}$$

The new iteration vectors satisfy the required condition for orthogonality:

$$[\varphi]^T[M][\hat{X}] = [0]. \tag{4.18}$$

However, in solving Equation (4.17), mass-orthogonality of the vectors in $[\hat{X}]$ to each other is violated. To evaluate the significance of this effect, consider the following. Define $[\alpha]$ as the mass-weighted projection of $[X]$ onto $[\varphi]$ prior to orthogonalization:

$$[\alpha] = [\varphi]^T[M][X]. \tag{4.19}$$

After orthogonalization by Equation (4.17), the new iteration vectors $[\hat{X}]$ satisfy Equation (4.18), however, they have been altered such that

$$[\hat{X}]^T[M][\hat{X}] = [I] + [\alpha]^T[\alpha]. \tag{4.20}$$

If the vector projections in $[\alpha]$ are of the order ϵ, the mass-weighted projections of the vectors in $[\hat{X}]$ on each other are on the order ϵ^2. Since the operation of Equation (4.17) is performed after every iteration, the projection values, ϵ, can be expected to be small. Thus, ϵ^2 will be smaller yet, and Equation (4.20) can be approximated by

$$[\hat{X}]^T[M][\hat{X}] = [I] \tag{4.21}$$

The vectors in $[\hat{X}]$ are used as $[X]$ in the next iteration without the need for each vector to be individually mass-orthogonalized to the others. While numerical values for the terms in $[\alpha]$ for various example problems have not been examined, the above orthogonalization procedure has not led to any stability or convergence difficulties.

4.6.3.4 Subspace Iteration with Hypermatrices

A summary of the subspace iteration method implemented in this study is presented in the following pseudo-code. The individual procedures are discussed in the following section.

```
     CALL INITIALIZE
     IF( SHIFT .NE. 0 ) CALL SHIFT_K
     CALL TRIANGULATE
     GO TO $TRANS
     LOOP
          IF( CONVERGENCE_COUNT .GT. 0 ) CALL ORTHOG_PHI
          CALL INERTIA_LOADS
          CALL LOAD_PASS
   $TRANS CALL TRANSFORM
          CALL JACOBI
          CALL NEW_X
          CALL TEST_CONVERGENCE
          IF( CONVERGE ) THEN
               CALL MOVE_PHI
               CALL REPLACE_X
               CALL UPDATE_ORTHOG
          END IF
          IF( ALL_CONVERGED ) EXIT
          IF( ITERATION LIMIT EXCEEDED ) EXIT
          CALL NEW_SHIFT
          IF( TIME_TO_SHIFT ) THEN
               CALL SHIFT_K
               CALL TRIANGULATE
          END IF
     END LOOP
```

4.6.3.5 Description of Procedures

Procedure INITIALIZE computes the subspace size, evaluates the discrete cosine functions used as initial iteration vectors, and initializes iteration variables. If the analyst has indicated that the structural model contains rigid body modes, variable SHIFT is set to a small negative value.

Procedure SHIFT_K applies the shift to the stiffness matrix which is stored in hypermatrix format. The shifted stiffness is

$$[K'] = [K] - SHIFT * [M]. \qquad (4.22)$$

Procedure TRIANGULATE performs Choleski decomposition on [K] if SHIFT equals zero or on [K'] if SHIFT is non-zero (see Equation (4.5)). During triangulation, the Sturm sequence check is performed. The number of negative terms that appear on the diagonal of [L] during decomposition identifies the number of eigenvalues below SHIFT. If the STURM CHECK property is specified by the analyst, this number is output during the solution of the eigenproblem.

Procedure ORGHOG_PHI performs Gram-Schmidt orthogonalization of the iteration vectors in [X] (see Equation (4.17)). The product $[\varphi][\varphi]^T[M]$ is computed by procedure UPDATE_ORTHOG prior to executing this procedure.

Procedure INERTIA_LOADS computes the inertia load vectors (see Equation (4.6)).

Procedure LOAD_PASS computes $[\tilde{X}]$ by performing a forward and a backward load-pass on the inertia loads (Equation (4.7)).

Procedure TRANSFORM computes the projected operators for the subspace. In the first iteration, $[\tilde{K}]$ is computed from

$$[\tilde{K}] = [\tilde{X}]^T[K][\tilde{X}]. \qquad (4.23)$$

In all other iterations, Equation (4.8) is used. $[\tilde{M}]$ is derived from Equation (4.9) in all iterations.

Procedure JACOBI solves the eigenproblem for the subspace, Equation (4.10). After solution of the eigenproblem, the eigenvalues $[\lambda]$ and the corresponding eigenvectors $[\Psi]$ are sorted in ascending order so that convergence of the vectors in [X] can be evaluated properly. In

Wilson's implementation [58], the shift is removed prior to solving the subspace equations:

$$([\bar{K}] + \text{SHIFT} * [M])\,[\Psi] = [\lambda][\bar{M}][\Psi]. \qquad (4.24)$$

Using this equation, the eigenvalues in $[\lambda]$ converge directly to the system eigenvalues $[\omega^2]$. If Equation (4.10) is solved, the eigenvalues in $[\lambda]$ differ from those of $[\omega^2]$ by SHIFT.

Procedure NEW_X computes the improved iteration vectors [X], Equation (4.11).

Procedure TEST_CONVERGENCE compares the values in $[\lambda]$ with those from the previous iteration. If the difference in λ_i from one iteration to the next is within the convergence tolerance (10^{-6} is typically used), that eigenvalue has converged. The sort in procedure JACOBI forces λ_1 to converge before λ_2, and so on. Therefore, convergence testing terminates with the first value that fails the test. If any values are found to converge, variable CONVERGE is set true, and the convergence counter is incremented. When the required number of eigenvalues has converged, variable ALL_CONVERGED is set true.

Procedure MOVE_PHI moves the converged eigenvectors from [X] into $[\varphi]$. The converged eigenvalues are moved from $[\lambda]$ to $[\omega^2]$.

Procedure REPLACE_X adds new iteration vectors to [X] to replace the converged eigenvectors. As the replacement vectors are generated, they are scaled by the largest eigenvalue estimate remaining in $[\lambda]$ to control overflow problems. The replacement vectors are then mass-orthogonalized to the other iteration vectors by Equation (4.13).

Procedure UPDATE_ORTHOG updates the product $[\varphi][\varphi]^T[M]$ to include the eigenvectors which have just converged. This product is updated each time an eigenvector converges. The product is then used in procedure ORTHOG_PHI to satisfy Equation (4.18). The objective is to minimize numerical operations; the alternative is to compute $[\varphi][\varphi]^T[M]$ for all eigenvectors each time procedure ORTHOG_PHI is executed. The drawback to this approach is that a fully populated, non-symmetric hypermatrix of the same order as [M] must be stored.

Procedure NEW_SHIFT determines if it is time to shift the stiffness matrix to improve convergence. If shifting is appropriate, variable TIME_TO_SHIFT is set true, and SHIFT is recomputed. The shifting strategy is based on the guidelines established in [58].

In procedure INITIALIZE, the iteration vectors are placed in $[\tilde{X}]$ rather than in [X]. The inertia-load and load-pass computations are skipped in the first iteration, and $[\tilde{X}]$ is used immediately in procedure TRANSFORM. This action serves two purposes. First, since the discrete cosine functions are not scaled to the physical characteristics of the structure, overflow problems are possible during the load-pass computations. Skipping INERTIA_LOADS and LOAD_PASS reduces exponent growth until the iteration vectors can be properly scaled in NEW_X. Second, since the initial iteration vectors are not mass-orthogonal, it is unwise to use them in inverse iteration (INERTIA_LOADS and LOAD_PASS). Performing the Ritz analysis (TRANSFORM, JACOBI, and NEW_X) serves to mass-orthogonalize the iteration vectors so that subsequent inverse iterations are stable.

4.7 Fixed-Interface Method

Since static condensation was functional in FINITE at the start of this study, extension of the system to support the fixed-interface method of modal synthesis was straightforward. The equations which define the condensed stiffness and mass matrices for a substructure, (2.10)-(2.16), provide the basis for the implementation. With the statically condensed stiffness matrix of the substructure available, the fixed-interface computations involve four steps:

1. Compute the static constraint modes $[\varphi^c]$,
2. Compute the Guyan reduced mass $[M^G]$,
3. Perform the fixed-fixed frequency analysis, and
4. Compute the mass coupling block $[M^{mn}]$.

At the conclusion of these operations, each of the individual components of the reduced stiffness and mass are computed separately. Assembly of the reduced matrices into the stiffness and mass matrices for the higher level structure is then performed as an independent function. These four steps along with the assembly procedure are discussed in detail below.

4.7.1 Static Constraint Modes

As defined in Chapter 2, a static constraint mode is the displaced configuration of the slave DOF resulting from a unit displacement applied to one of the master DOF while all other master DOF are held fixed. Equation (2.5) suggests that the static constraint modes can be computed by standard equation solving techniques:

$$[K^{ss}][\varphi^c] = -[K^{sm}]. \tag{2.5}$$

However, it is not necessary to perform both the forward and backward load-pass operations. Only a special back-pass is required as described in the following.

The procedure used in FINITE for static condensation involves "partial decomposition" [57] of the stiffness matrix. Consider the stiffness matrix for a substructure which is to be condensed. Partitioning the matrix to separate master and slave DOF yields

$$[K] = \left[\begin{array}{c|c} K^{ss} & K^{sm} \\ \hline K^{ms} & K^{mm} \end{array}\right] \tag{4.25}$$

where the superscripts on the submatrices denote master (m) and slave (s) DOF. Choleski decomposition is applied to completely eliminate the slave DOF in $[K^{ss}]$. Similarly, the master-slave coupling terms in $[K^{ms}]$ are reduced following the standard procedures for off-diagonal terms. A partial decomposition is then performed on the $[K^{mm}]$ submatrix of master DOF coefficients to eliminate the coupling effect of the slave DOF in submatrix $[K^{ms}]$. The modified submatrix $[K^{mm}]$ becomes the desired condensed stiffness matrix for the substructure. In partitioned form, the partially decomposed stiffness matrix becomes

$$[L^{p}] = \left[\begin{array}{c|c} L^{ss} & 0 \\ \hline Y^{T} & K^{G} \end{array}\right] \tag{4.26}$$

where $[K^{G}]$ is the statically condensed (or Guyan reduced) stiffness, $[L^{ss}]$ is the lower triangular Choleski factor of the $[K^{ss}]$, and [Y] is the matrix of "partial static constraint modes." As a consequence of

the condensation process, the submatrix [Y] contains the result of a standard forward substitution:

$$[L^{ss}][Y] = [K^{sm}]. \tag{4.27}$$

To complete the static constraint modes $[\varphi^c]$, only a backward substitution is necessary:

$$[L^{ss}]^T[\varphi^c] = -[Y]. \tag{4.28}$$

Implementation of this backward substitution function required a minor addition to subsystem TRIANGULATE. TRIANGULATE is invoked by subsystem ASSEMBLER when stiffness and mass matrix assembly requires condensation of lower level substructures. After the condensed stiffness is computed as described above, subsystem LOADPASS is initiated by TRIANGULATE to perform the backward substitution needed to complete the static constraint modes. The matrix $[\varphi^c]$ is then stored in the SOLVER database and mass matrix condensation begins.

4.7.2 Guyan Reduced Mass

The second step in the condensation process is the computation of the Guyan reduced mass. This procedure is implemented in subsystem TRIANGULATE directly as defined by Equation (2.15) for a consistent mass formulation and Equation (2.16) for lumped mass models. Repeating those equations for reference:

$$[M^G] = [M^{mm}] + [\varphi^c]^T[M^{ss}][\varphi^c] + [\varphi^c]^T[M^{sm}] + [M^{ms}][\varphi^c] \tag{2.15}$$

$$[M^G] = [M^{mm}] + [\varphi^c]^T[M^{ss}][\varphi^c] \tag{2.16}$$

The algorithm for hypermatrix triple products described earlier in this chapter at first appears to have application in computing $[M^G]$.

However, in computing the mass coupling block, $[M^{mn}]$, it is more economical (fewer numerical operations are required) to use the conventional procedure for computing triple products. The matrix product $[\varphi^c]^T[M^{ss}]$ is used in computation of both $[M^G]$ and $[M^{mn}]$. Therefore, it is more efficient to compute the product once and hold it as a temporary matrix, [T]. Then [T] is used in Equation (2.15) or (2.16) to compute $[M^G]$ and again later to compute $[M^{mn}]$.

One additional facet of this step needs discussion. For consistent mass formulations, the off-diagonal submatrices, $[M^{sm}]$ and $[M^{ms}]$ are included in the computation of $[M^G]$. Since the mass matrix is symmetric:

$$[M^{ms}][\varphi^c] = ([\varphi^c]^T[M^{sm}])^T \tag{4.29}$$

and only the matrix product $[M^{ms}][\varphi^c]$ must be computed. The other product is obtained by simple transposition.

When $[M^G]$ is finally computed, it too is stored in the SOLVER database.

4.7.3 Fixed-Fixed Frequency Analysis

The normal modes used in the fixed-interface method are defined by the eigenvalue problem:

$$[K^{ss}][\bar{\varphi}^n] = [\hat{\omega}^2][M^{ss}][\bar{\varphi}^n]. \tag{4.30}$$

Solution of this problem for the selected frequencies and mode shapes is performed by subsystem EIGEN as described in Section 4.6. Constraint of the master DOF implied by Equation (4.30) is provided through equation

partitioning. Since the slave DOF are blocked in the top rows and columns of the stiffness and mass matrices, the master DOF are effectively constrained during frequency analysis by ignoring entries in [K] and [M] below the last slave DOF. After solution, both the matrix of normal modes, $[\bar{\varphi}^n]$, and the associated frequencies, $[\hat{\omega}^2]$, are saved in the SOLVER database. The normal modes are used in computation of the mass coupling block and the frequencies represent the normal stiffness coefficients in the reduced stiffness matrix.

While the fixed-fixed frequency analysis is logically the third step in the reduction procedure, implementation followed a different scheme. This step is actually performed before the other three steps. In subsystem ASSEMBLER, the need for fixed-fixed normal modes is determined prior to invoking subsystem TRIANGULATE. If normal modes are used in condensation, subsystem EIGEN is called first. Upon return from EIGEN, ASSEMBLER initiates subsystem TRIANGULATE to do the condensation. Once TRIANGULATE is initiated, steps 1, 2, and 4 are completed without interruption because the fixed-fixed eigenpairs are already available.

4.7.4 Mass Coupling Block

The off-diagonal submatrix in the reduced mass matrix, $[M^{mn}]$, contains the coupling terms between the normal and the master DOF of the substructure. The submatrix is defined by Equation (2.12) for consistent mass models and by Equation (2.13) when a lumped mass formulation is used. Those equations are:

$$[M^{mn}] = [M^{ms}][\bar{\varphi}^n] + [\varphi^c]^T[M^{ss}][\bar{\varphi}^n] \tag{2.12}$$

$$[M^{mn}] = [\varphi^c]^T[M^{ss}][\bar{\varphi}^n] \tag{2.13}$$

For the lumped mass formulation, Equation (2.13) is computed by a standard matrix product using the temporary matrix [T] as described above. When a consistent mass is used, Equation (2.12) is rearranged so that only one matrix product is computed. The off-diagonal block $[M^{ms}]$ is first added to [T] and then this sum is post-multiplied by $[\bar{\varphi}^{n}]$. The computations actually take the form:

$$[M^{mn}] = ([M^{ms}] + [T]) * [\bar{\varphi}^{n}] \qquad (4.31)$$

where $[T] = [\varphi^{c}][M^{ss}]$.

4.7.5 Assembly of the Reduced Stiffness and Mass Matrices

When subsystem TRIANGULATE terminates execution after performing the above reduction, the reduced stiffness and mass matrices are actually broken into four components, each stored separately in the SOLVER database. The components are $[K^{G}]$ and $[\hat{\omega}^{2}]$ which form the reduced stiffness and $[M^{G}]$ and $[M^{mn}]$ which form the reduced mass. Subsystem ASSEMBLER retrieves these components from the SOLVER database and assembles them into the reduced stiffness and mass matrices. Assembly occurs when the actual matrices are needed to form the stiffness and mass for a higher level structure.

4.8 Restart and Reanalysis

Prior to performing the structural analysis, an analyst does not generally know the number of natural frequencies below a certain target frequency or the number of iterations required to compute a specified number of eigenpairs. For substructured models, the analyst must also

select the number of normal DOF to retain in each condensed substructure. If too few normal DOF are selected, overall response of the structural model will be degraded. If too many normal DOF are selected, the reduction process becomes excessively expensive. Selection of the "correct" number of DOF to retain is based on experience and judgement. However, even experienced analysts can seldom anticipate the number of normal DOF needed for accurate and economical solution of a new structural model. Analysis software must provide the capabilities for the analyst to gain this knowledge in an iterative fashion. In order to efficiently achieve such an iterative solution, the software must support automatic restart and partial reanalysis.

Automatic restart is defined as the resumption of a previously terminated analysis without loss of computed results. For example, suppose that an analyst computes the first 25 frequencies and mode shapes for a structure and requests output of the natural frequencies but terminates execution of the analysis prior to obtaining mode shape output. Automatic restart allows access to the existing databases for output of the mode shapes without recomputing them.

Partial reanalysis is the ability to make modifications to a structural model and to recompute the response of the highest level structure without completely reanalyzing the entire structural model. For example, suppose that a structure with three condensed substructures has been analyzed and the analyst wants to refine the definition of the first substructure. A partial reanalysis involves restarting the fixed-fixed frequency analysis of that substructure, computing additional normal DOF, recondensing the substructure, assembling it into the highest level structure, and reanalyzing the highest level structure

without repeating the condensation and assembly of the two unmodified substructures. This capability of the software is critical to the success of the analysis of multilevel substructured models in which fixed-interface reduction is used throughout the hierarchy of the structural model.

Implementation of a general restart and reanalysis capability is much more complex than the computational procedure indicates (see Section 2.2.3). The reason is that the critical procedures are not computational. Instead, extensive changes in both size and content of previously created data structures are required. Sophisticated data management procedures are the prerequisite for successful restart and reanalysis. To begin the reanalysis, a complex traversal of the structural hierarchy is required to validate (or invalidate) existing data, to determine what needed data are missing, and to determine the effects of invalid or missing data at each level of the hierarchy. Once this traversal is complete, the reanalysis begins. Existing valid data is used wherever possible. New computations are performed only when necessary.

4.8.1 Automatic Restart

Automatic restart was an operational feature of FINITE at the start of this study. After termination of an analysis, the existing databases could be accessed again and any conventional request issued. This includes definition and displacement computation for a new static loading condition, output of previously computed displacements, strains, or stresses for a structure, and continuation of a nonlinear static analysis. The new dynamic analysis features are also implemented with

restart capabilities, the most powerful of which is frequency analysis restart. Frequency analysis restart involves continuation of a previous frequency analysis to compute additional eigenpairs for any specified structure, at any level of the structural hierarchy. Since the generalized Jacobi method yields all eigenpairs for a structure, frequency analysis restart applies only to subspace iteration.

The analyst defines restart of subspace iteration by specifying the number of additional eigenpairs to compute and a value for the initial subspace shift. The initial shift is some value greater than the last converged eigenvalue but less than an estimate for the next eigenvalue in the spectrum. For example, suppose that in the first analysis run, 15 eigenpairs converged with the largest eigenvalue equal to 2.5E+06. When this initial run terminates, FINITE outputs an estimate for eigenvalue number 16, say 4.2E+06. If the analyst wants a total of 20 eigenvalues for the structure, parameters for restart of subspace iteration would be defined as follows:

```
FREQUENCY ANALYSIS TYPE SUBSPACE
     PROPS  NUMBER OF PAIRS  5  ITERATIONS 10,
            MINIMUM FREQUENCY  3.3E+06
```

In the above the MINIMUM FREQUENCY is the value to which a shift is applied before continuing the analysis.

The key to efficient restart of subspace iteration is the re-use of the previous set of iteration vectors. When the initial analysis run terminates, several of the vectors in the iteration set will be nearly converged. (This is the basis for the estimate of eigenvalue number 16 in the above example.) Since these vectors are the best known estimates for the real eigenvectors, they provide the optimum set of initial iteration vectors. Therefore, it is imperative that the software system

make these vectors available for re-use. Complications for data management arise when the analyst changes another property of the analysis method: the subspace size. Such a change forces the hypermatrix that stores the iteration vectors to be resized (columns are either added or removed depending on an increase or decrease of the subspace size). If the subspace size is increased, new "cosine-function" iteration vectors are added to fill out the set.

Another major task performed prior to restarting the subspace computations is moving the existing eigenvectors into the SOLVER database and storing them in hypermatrix form. The eigenvectors are needed for the orthogonalization of iteration vectors after each iteration. After these two data management operations are performed, the frequency analysis is resumed. It is important to note that these data handling tasks are performed automatically and are transparent to the analyst. The analyst's contribution to restart is simply the selection of the number of additional eigenpairs and the specification of an initial shift. Since very few numerical operations are performed during this set-up phase, overhead for analysis restart is minimal.

4.8.2 Partial Reanalysis

As discussed in Chapter 2, an analyst often requires reanalysis of a model as a check on the quality of the reduction of one or more substructures. To obtain the check, additional normal DOF are added to selected substructures and the analysis is repeated.

For efficient restart, computations must be limited to only those portions of the model affected by the modifications. Reanalysis begins with the computation of additional fixed-fixed normal modes for the

substructures in question. When subspace iteration is specified for the frequency analysis, restart is initiated as described in the previous section. The tables which store the frequencies $[\hat{\omega}^2]$ and mode shapes $[\bar{\varphi}^n]$ are resized (enlarged) for storage of the newly computed data. After the additional eigenpairs are determined, they are stored with their counterparts from the previous analysis.

The next step is to compute a new mass coupling block $[M^{mn}]$ for the substructure. The new mass coupling block contains one new column for each new mode shape in $[\bar{\varphi}^n]$, with the existing columns remaining unchanged. Therefore, it is sufficient just to resize the matrix $[M^{mn}]$ and compute the new columns by the procedure discussed in Section 4.7.4.

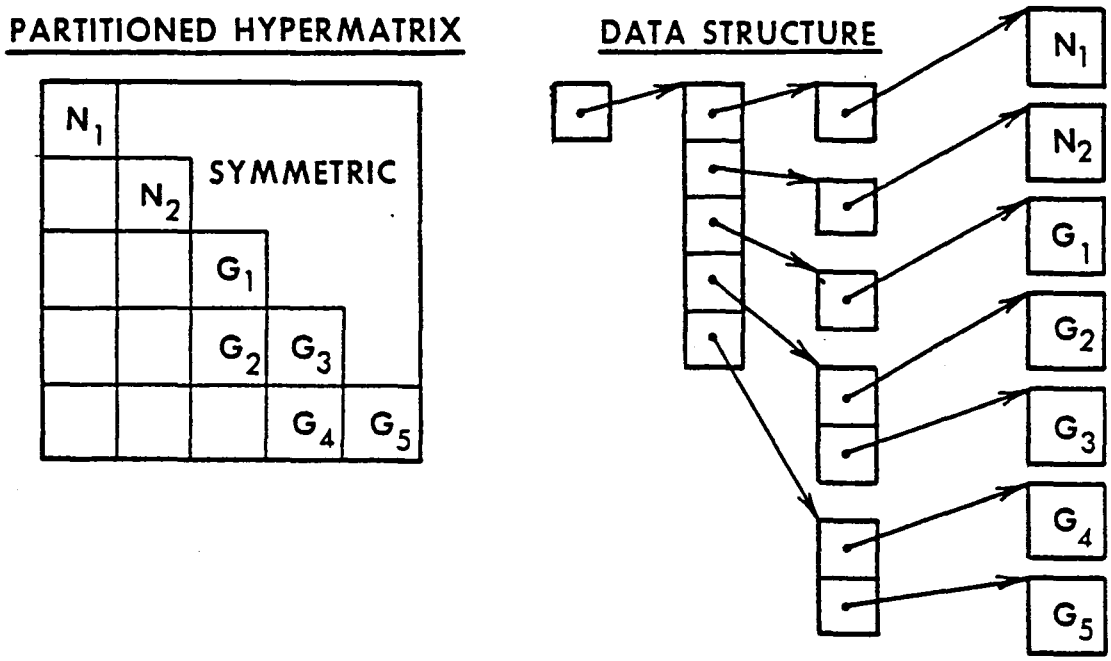
The most complex step in the implementation is assembly of the structure stiffness and mass matrices in which the reanalyzed substructures are used. The reanalysis procedure adds additional normal DOF to the condensed substructures. The geometric DOF are not affected. Therefore, when these substructures are re-assembled into the next level of the hierarchy, only the normal DOF are processed. The complication arises in reorganizing the hypermatrices that hold the stiffness and mass at the higher levels.

Since the normal DOF are located at the top of the coefficient matrices, the geometric DOF must be shifted down in the tables as new normal DOF are added. Rather than move actual blocks of numerical data, it is more efficient to create a new pointer hierarchy for the table and then swap pointers from the old to the new.

Figure 4.10 illustrates the procedure for resizing the stiffness matrix. Suppose that the existing stiffness is partitioned into 5 hyperrows and 5 hypercolumns, with the first 2 hyperrows and hypercolumns allocated to the normal DOF. Two non-zero submatrices (N_1 and N_2) are used for the normal DOF and 5 for the geometric DOF (G_1 - G_5). With the addition of new normal DOF to the lower level substructures, a new hyperrow and hypercolumn is added to contain the 3 normal DOF submatrices. Rather than create an entirely new hierarchy to store the expanded matrix, a new set of pointer vectors is created. Pointers to the individual geometric DOF submatrices, G_1 - G_5, are copied into the new pointer hierarchy and the old pointer structure is destroyed. Actual submatrices are not moved. At this point the new normal DOF submatrices, $\bar{N}_1$ - $\bar{N}_3$, are assembled from existing and newly added data.

Resizing and re-assembly of the structure mass matrix follows a similar procedure. Submatrices containing only geometric DOF are retained without change and submatrices containing normal DOF are completely re-assembled after the new DOF are added.

A. ORIGINAL STIFFNESS MATRIX



B. RESIZED AND RE-ASSEMBLED STIFFNESS MATRIX

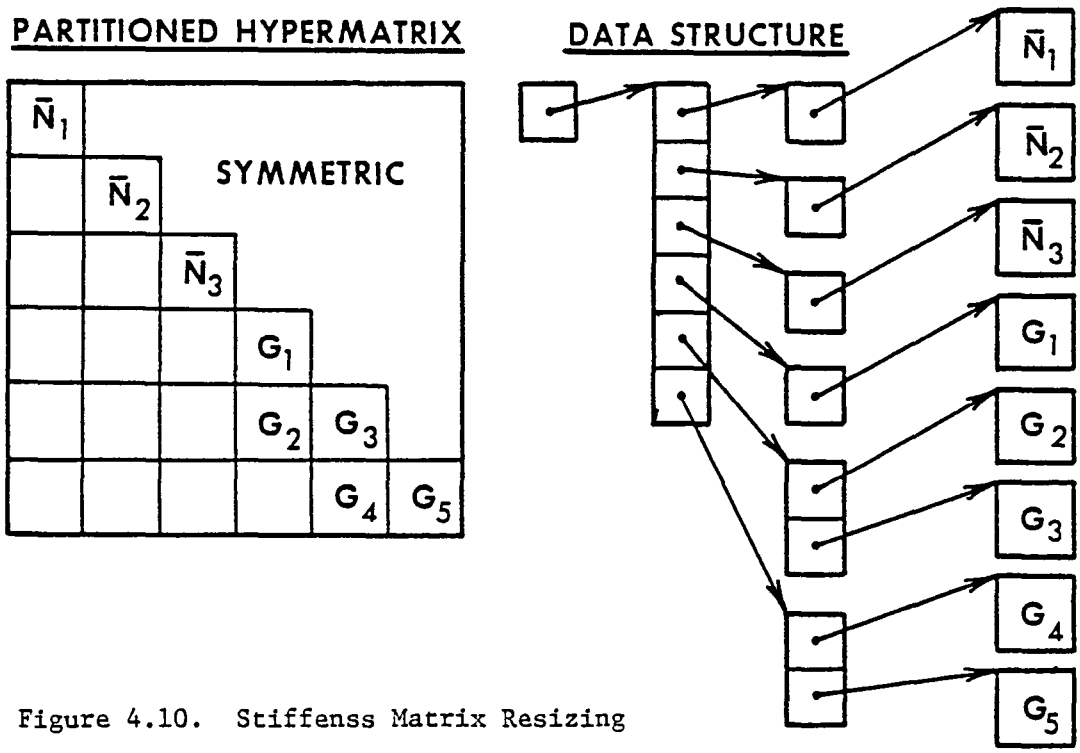


Figure 4.10. Stiffenss Matrix Resizing

CHAPTER 5 -- NUMERICAL EXAMPLES

5.1 General

The modeling and analysis procedures developed in this study are demonstrated and evaluated in this chapter. Numerical studies on example structures are performed to demonstrate two principal products of this research. First, the feasibility of multilevel substructured analysis using modal synthesis techniques in a general purpose software system is considered. Preliminary studies of solution accuracy and computational efficiency are made to demonstrate the advantages of the numerical procedures. Second, unique features of the software are demonstrated. The convenience of the flexible user interface, automatic restart, and partial reanalysis are all illustrated.

Natural frequencies, mode shapes, and modal strains are computed for both substructured and non-substructured models. Each example structure is initially modeled and analyzed without substructuring to establish a baseline against which approximate results are compared. Subsequent analyses are performed on the substructured models with varying topology and degrees of reduction.

The first example involves the analysis of a cantilever box structure composed of flat shell elements. This example demonstrates the performance of the fixed-interface method applied to multilevel substructured models. Both computational effort and solution accuracy are evaluated. Detailed comparisons of natural frequencies, mode shapes, and modal strains are made for this example.

The second example illustrates restart, reanalysis, and the capabilities of the software to process rigid-body modes. Three-dimensional truss elements are used to model a structure which has the shape of a double tetrahedron. Emphasis in this example is placed on the user interface and restart capabilities. Only frequencies are considered in the accuracy comparisons.

All numerical computations were performed on a Harris 500 computer. On this machine, floating point numbers are represented with a 38 bit mantissa and a 7 bit exponent. This format represents numerical values which vary in magnitude from 10^{-38} to 10^{+39} with 11 - 12 decimal digits of precision.

5.2 Cantilever Box

The first example structure is a thin-walled, cantilever box, open on the top as shown in Figure 5.1. The structure is modeled with flat-shell elements derived from plate and membrane elements. At nodes in which connecting elements are not coplanar, there are six active DOF (three translations and three rotations). At nodes in which elements are coplanar, the rotation normal to the plane is constrained leaving only five active DOF at the node. All analyses of this structure incorporate a consistent mass formulation.

The box structure is analyzed using three different models. The first model is not substructured and contains 172 flat shell elements and 196 nodes. This model, named BOX_1, provides the baseline against which the approximate results of the substructured models are compared. The finite element mesh for structure BOX_1 is shown in Figure 5.2. Input data to generate the mesh and to perform the analysis are shown in

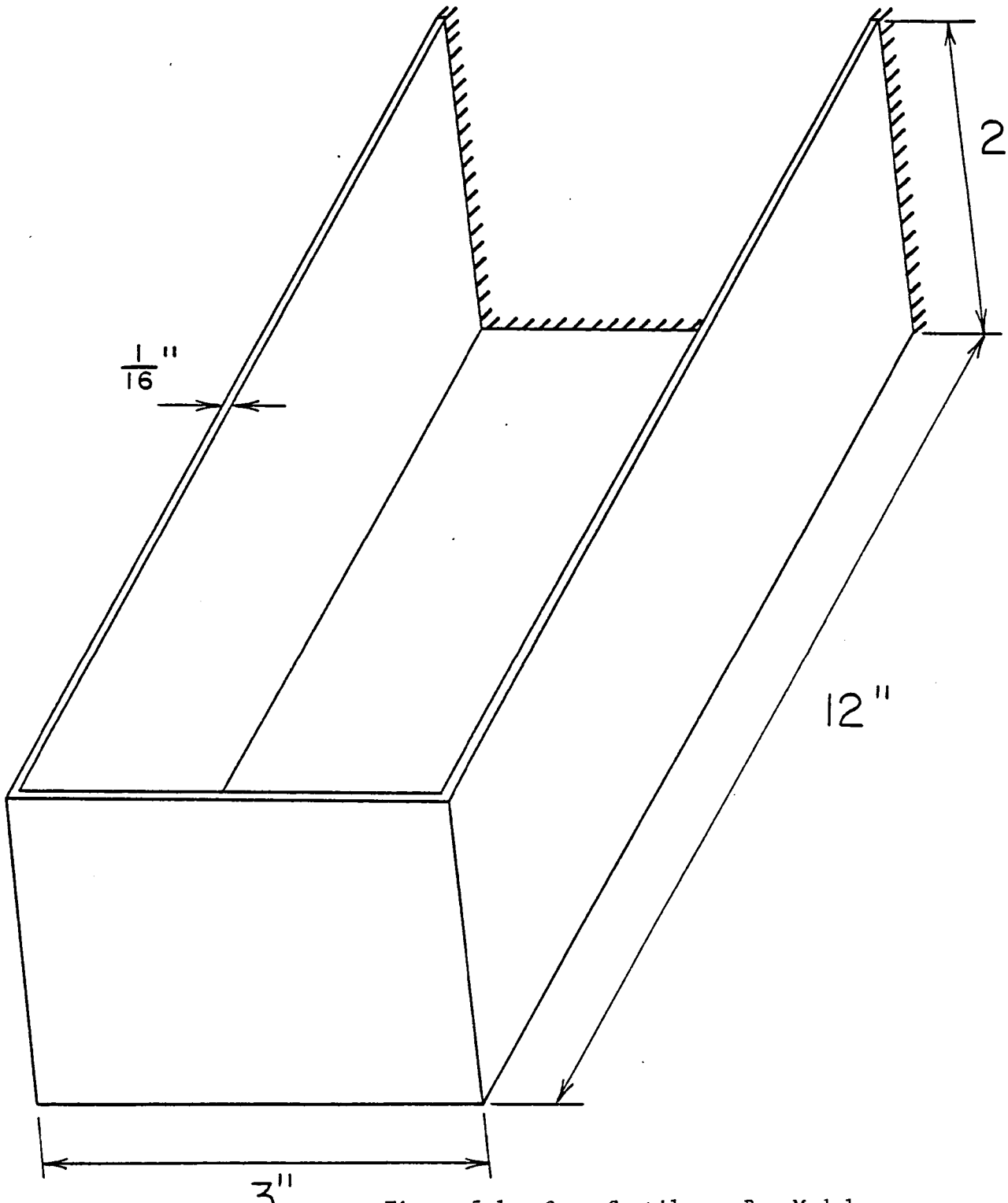


Figure 5.1. Open Cantilever Box Model

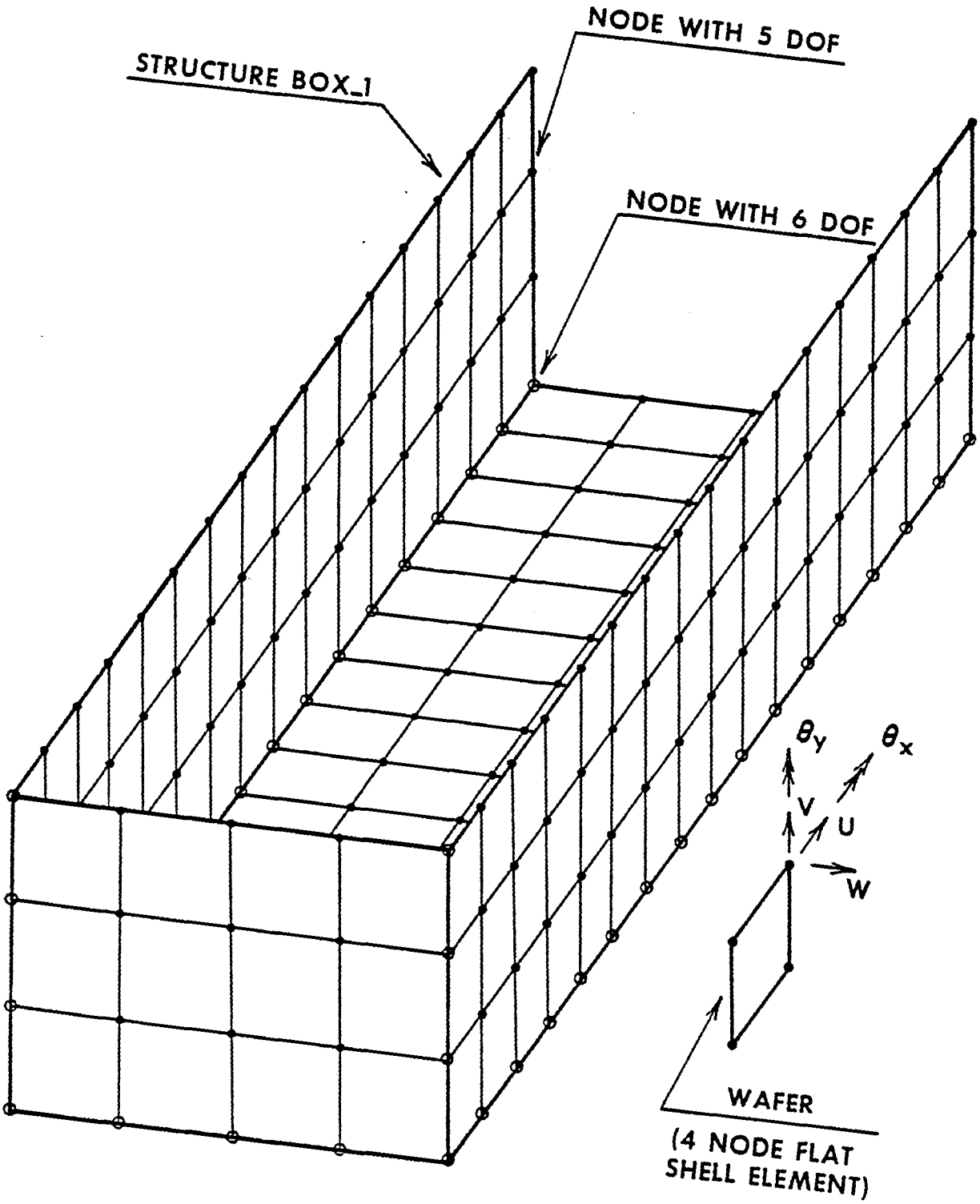


Figure 5.2. Finite Element Mesh for Structure BOX_1

Figure 5.3. Since each shell element in the model is identical to all the others, except for orientation, a single "stand-alone" element named WAFER is defined first. The stiffness and mass matrices for this element are computed only once and then are used repeatedly for each occurrence of WAFER in structure BOX_1. In order to extend the definition of the model from static to dynamic analysis, only two additions to the input are made. First the mass of element WAFER is defined. A CONSISTENT mass formulation is chosen with a MASS_DENSITY of 7.339E-04. Then the frequency analysis method is selected. Subspace iteration is used to evaluate the first 10 natural frequencies and mode shapes for the structure.

The second model, structure BOX_2, uses one level of substructuring with condensation to reduce the number of DOF which are present in the highest level structure. The mesh for this model is illustrated in Figure 5.4 and the POL input is shown in Figure 5.5. The hierarchy of the structural model is shown in Figure 5.6. The first level of substructures contains the parent structures: structure SIDE (a side panel) and structure BOTTOM (a bottom panel). The condensed version (child) of each of these substructures contains the boundary nodes from the parent structure and a selected number of normal DOF. Normal DOF are computed by a fixed-fixed vibration analysis of the parent. The condensation procedure is specified in the definition of the child structures SIDE_CON and BOTT_CON. The highest level structure, BOX_2, has only 13 elements and 79 nodes (plus the normal DOF retained during condensation).

Figure 5.7 illustrates the third model of the cantilever box structure, BOX_3. This model contains two levels of substructuring. Input

```
*RUN FINITE
C
C           OPEN CANTILEVER BOX STRUCTURE USED TO DEMONSTRATE THE
C           PERFORMANCE OF THE FIXED-INTERFACE METHOD WITH
C           MULTILEVEL SUBSTRUCTURED MODELS.
C
C           THE STRUCTURE USES 172 RFSHELL ELEMENTS FORMED INTO A
C           LONG STEEL BOX WHICH IS OPEN ON TOP AND CANTILEVERED AT
C           ONE END.  THE BOX IS  3.0" WIDE, 2.25" HIGH, AND 12.0"
C           LONG WITH CONSTANT WALL THICKNESS OF  0.0625".
C
C           THIS IS THE    NON-SUBSTRUCTURED   VERSION OF THE MODEL.
C                          ================
C
  ELEMENT WAFER
    TYPE RFSHELL CONSISTENT E 30000. NU 0.3 THICKNESS .0625 ,
         SHORT OUTPUT    MASS_DENSITY .0007339  NOSPRINGS
    COORDINATES
      1    0.0    0.0
      2    0.75   0.0
      3    0.75   0.75
      4    0.0    0.75
C
C
  STRUCTURE BOX_1
    NUMBER OF ELEMENTS 172  NODES 196
    ELEMENTS ALL TYPE WAFER  ROTATION BY COORDINATES
C
    COORDINATES
      1      0.0      2.25     0.0
      4      0.0      0.0      0.0
      8      3.0      0.0      0.0
     11      3.0      2.25     0.0
    177      0.0      2.25    12.0
    180      0.0      0.0     12.0
    184      3.0      0.0     12.0
    187      3.0      2.25    12.0
    GEN  1-4  IN X 1-177 BY 11 IN Y
    GEN  4-8  IN X 4-180 BY 11 IN Y
    GEN  8-11 IN X 8-184 BY 11 IN Y
    GEN  177 188 189 190 187
    GEN  178 191 192 193 186
    GEN  179 194 195 196 185
C
C
    INCIDENCES
C
      GEN 10 IN X 16 IN Y AS 1-160 FROM 1 2 13 12 ADD 1 IN X 11 IN Y
C
      161  177  178  191  188
      162  188  191  192  189
      163  189  192  193  190
      164  190  193  186  187
      165  178  179  194  191
      166  191  194  195  192
      167  192  195  196  193
      168  193  196  185  186
      169  179  180  181  194
      170  194  181  182  195
      171  195  182  183  196
      172  196  183  184  185
C
    CONSTRAINTS
C
C             FIX THE NODES AT THE CANTILEVER WALL.
C
      1-11 ALL = 0.0
C
C             FIX THE THETA DOF WITH ZERO OUT-OF-PLANE STIFFNESS.
C
      12-166 BY 11, 13-167 BY 11, 14-168 BY 11    THETAX  =  0.0
      16-170 BY 11, 17-171 BY 11, 18-172 BY 11    THETAY  =  0.0
      20-174 BY 11, 21-175 BY 11, 22-176 BY 11    THETAX  =  0.0
      188-196                                     THETAZ  =  0.0
C
    FREQUENCY ANALYSIS TYPE SUBSPACE
      PROPERTIES NUM PAIRS 10 ITERATIONS 40 STURM CHECK
C
    COMPUTE NATURAL FREQUENCIES
    OUTPUT  NATURAL FREQUENCIES  MODE SHAPES
    STOP
```

Figure 5.3. POL Definition of Structure BOX_1

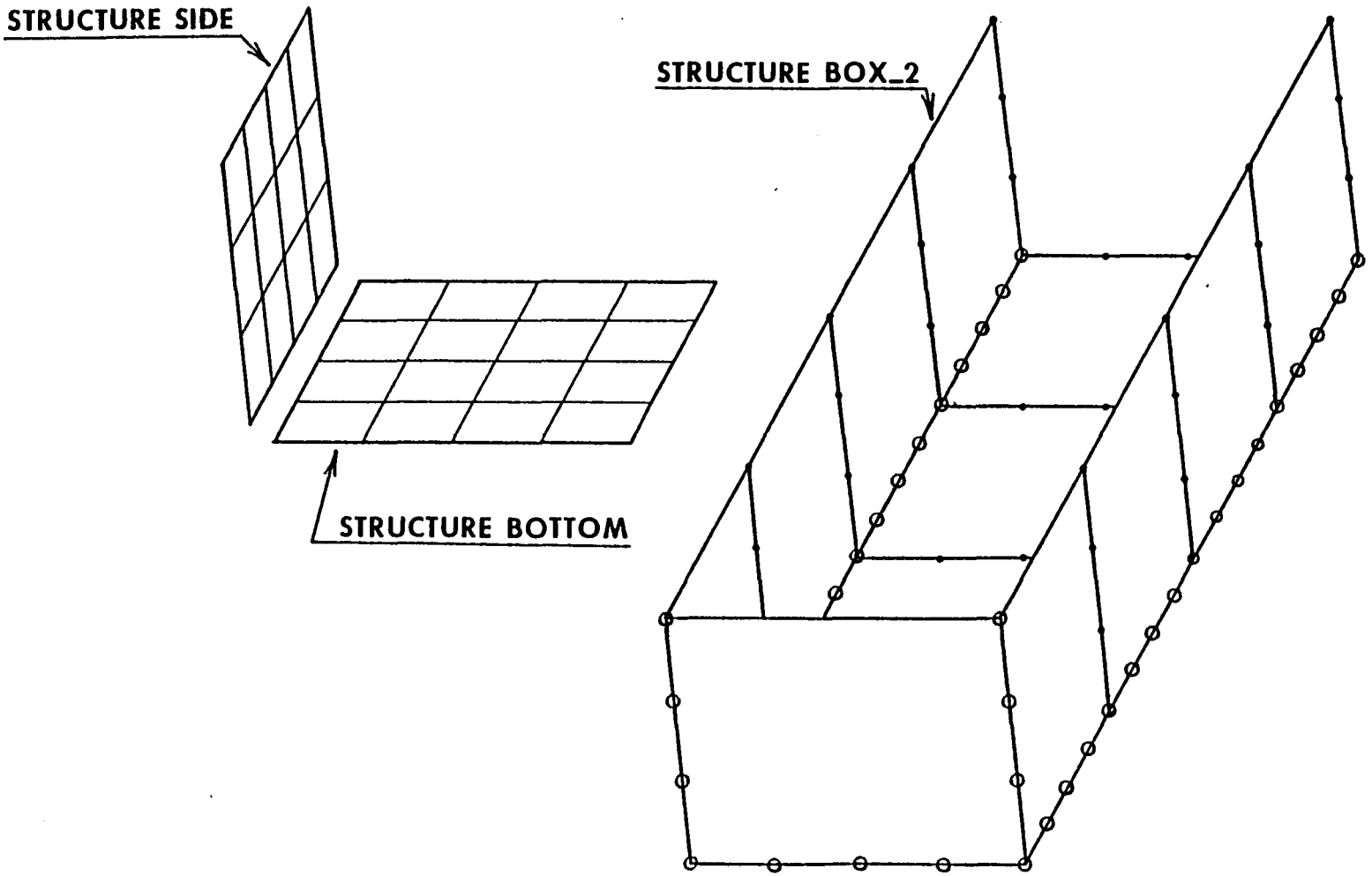


Figure 5.4. Finite Element Mesh for Structure BOX_2

```
*RUN FINITE
C          MODEL 2B:  SUBSTRUCTURED VERSION OF THE CANTILEVER BOX
C                     MODEL.  SUBSTRUCTURES ARE REDUCED BY THE
C                     FIXED-INTERFACE METHOD.  THERE ARE 5 NORMAL
C                     DOF RETAINED IN EACH SUBSTRUCTURE.
C                     THIS MODEL USES ONE LEVEL OF SUBSTRUCTURING.
C                                 =========
  ELEMENT WAFER
    TYPE RFSHELL CONSISTENT  E 30000. NU 0.3  THICKNESS 0.0625,
          SHORT OUTPUT       MASS_DENSITY 0.0007339  NOSPRINGS
C
    COORDINATES
      1   0.0   0.0
      2   0.75  0.0
      3   0.75  0.75
      4   0.0   0.75
C
  STRUCTURE SIDE
    NUMBER OF ELEMENTS 12  NODES 20
    ELEMENTS ALL TYPE WAFER  ROTATION BY COORDINATES
C
    COORDINATES
       1      0.0      2.25     0.0
       4      0.0      0.0      0.0
      17      0.0      2.25     3.0
      20      0.0      0.0      3.0
      GEN 1-4 IN X 1-17 BY 4 IN Y
C
    INCIDENCES
      GEN 3 IN X 4 IN Y AS 1-12 FROM 1 2 6 5 ADD 1 IN X 4 IN Y
C
    FREQUENCY ANALYSIS TYPE SUBSPACE
      PROPERTIES NUM PAIRS 5 ITERATIONS 15 STURM CHECK
C
    CONSTRAINTS
       5-7, 9-11, 13-15  THETAX = 0.0
C
  STRUCTURE SIDE_CON
    NUMBER OF ELEMENTS 1 NODES 11
    ELEMENT 1 TYPE SIDE CONDENSED   RETAIN NORMAL 1-5
C
    INCIDENCES
      1  1-4, 8, 12, 16, 20, 19, 18, 17
C
  STRUCTURE BOTTOM
    NUMBER OF ELEMENTS 16 NODES 25
    ELEMENTS ALL TYPE WAFER  ROTATION BY COORDINATES
C
    COORDINATES
       1   0.0   0.0   0.0
       5   3.0   0.0   0.0
      21   0.0   0.0   3.0
      25   3.0   0.0   3.0
      GEN 1-5 IN X 1-21 BY 5 IN Y
C
    INCIDENCES
      GEN 4 IN X 4 IN Y AS 1-16 FROM 1 2 7 6 ADD 1 IN X 5 IN Y
C
    CONSTRAINTS
       7-9, 12-14, 17-19  THETAY  =  0.0
C
    FREQUENCY ANALYSIS TYPE SUBSPACE
      PROPS  NUM PAIRS 5 ITERATIONS 15 STURM CHECK
C
  STRUCTURE BOTT_CON
    NUMBER OF ELEMENTS 1 NODES 16
    ELEMENT 1 TYPE BOTTOM  CONDENSED   RETAIN NORMAL 1-5
C
    INCIDENCES
      1   1-5 10 15 20 25 24 23 22 21 16 11 6
C
  STRUCTURE BOX_2
    NUMBER OF NODES 79 ELEMENTS 13
    ELEMENTS
      1-8  TYPE SIDE_CON  ROTATION SUPPRESSED
      9-12 TYPE BOTT_CON  ROTATION SUPPRESSED
        13 TYPE SIDE_CON  ROTATION Y  90.0
C
    INCIDENCES
      GEN 1-4 FROM 1 2 3 4 12 13 14 21 20 19 18 ADD 17
      GEN 5-8 FROM 11 10 9 8 15 16 17 25 26 27 28 ADD 17
      GEN 9-12 FROM 4-8 15-17 25-21 BY -1 14 13 12 ADD 17
      13  69-79
C
    CONSTRAINTS
        1-11  ALL = 0.0
        18-20  26-28  35-37  43-45  52-54  60-62  THETAX  =  0.0
        22-24  39-41  56-58                       THETAY  =  0.0
C
    FREQUENCY ANALYSIS TYPE SUBSPACE
      PROPS NUM PAIRS 10 ITERATIONS 30   STURM CHECK
C
    COMPUTE NATURAL FREQUENCIES
    OUTPUT  NATURAL FREQUENCIES    MODE SHAPES
    STOP
```

Figure 5.5. POL Definition of Structure BOX_2

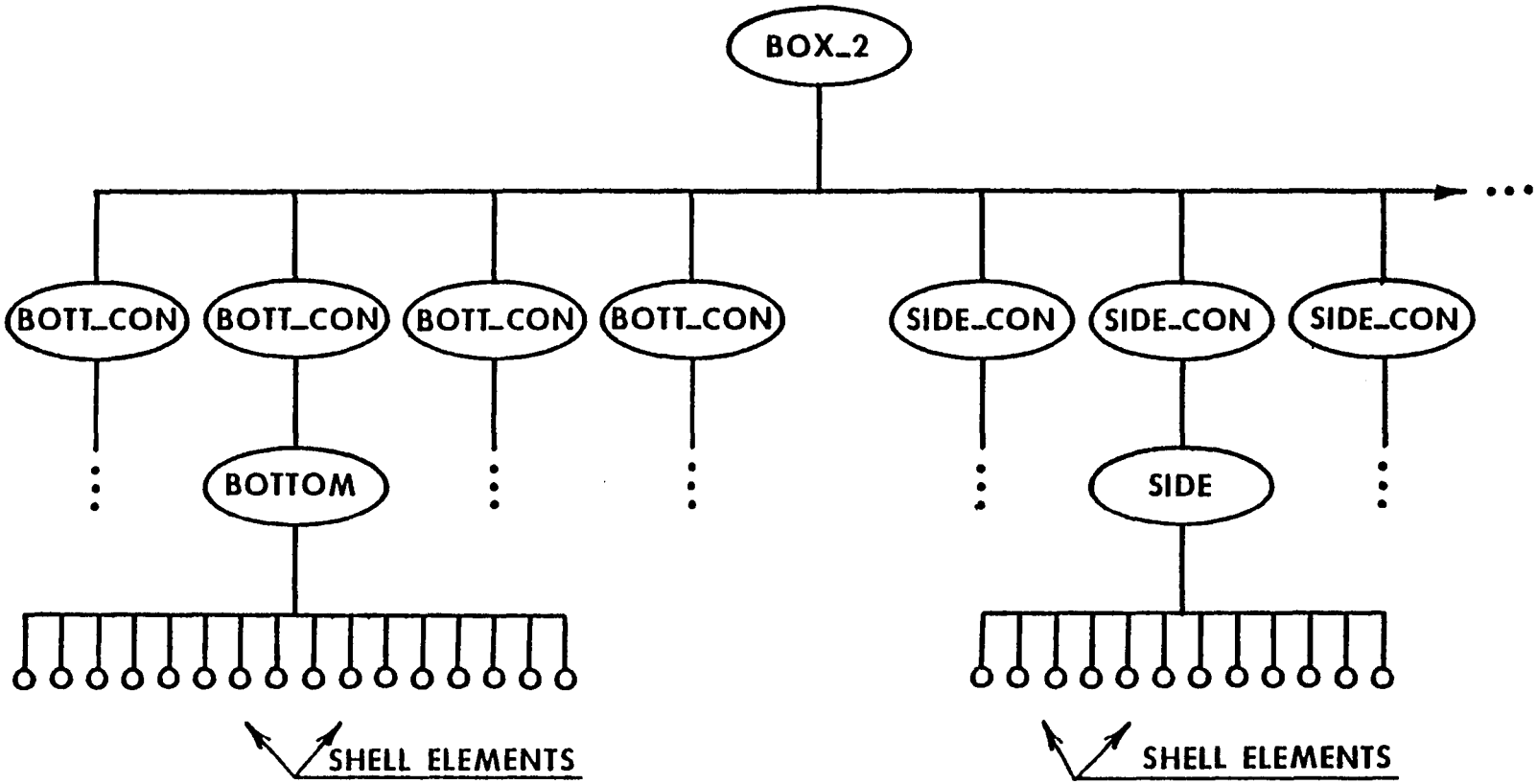


Figure 5.6. Hierarchy of Structure BOX_2

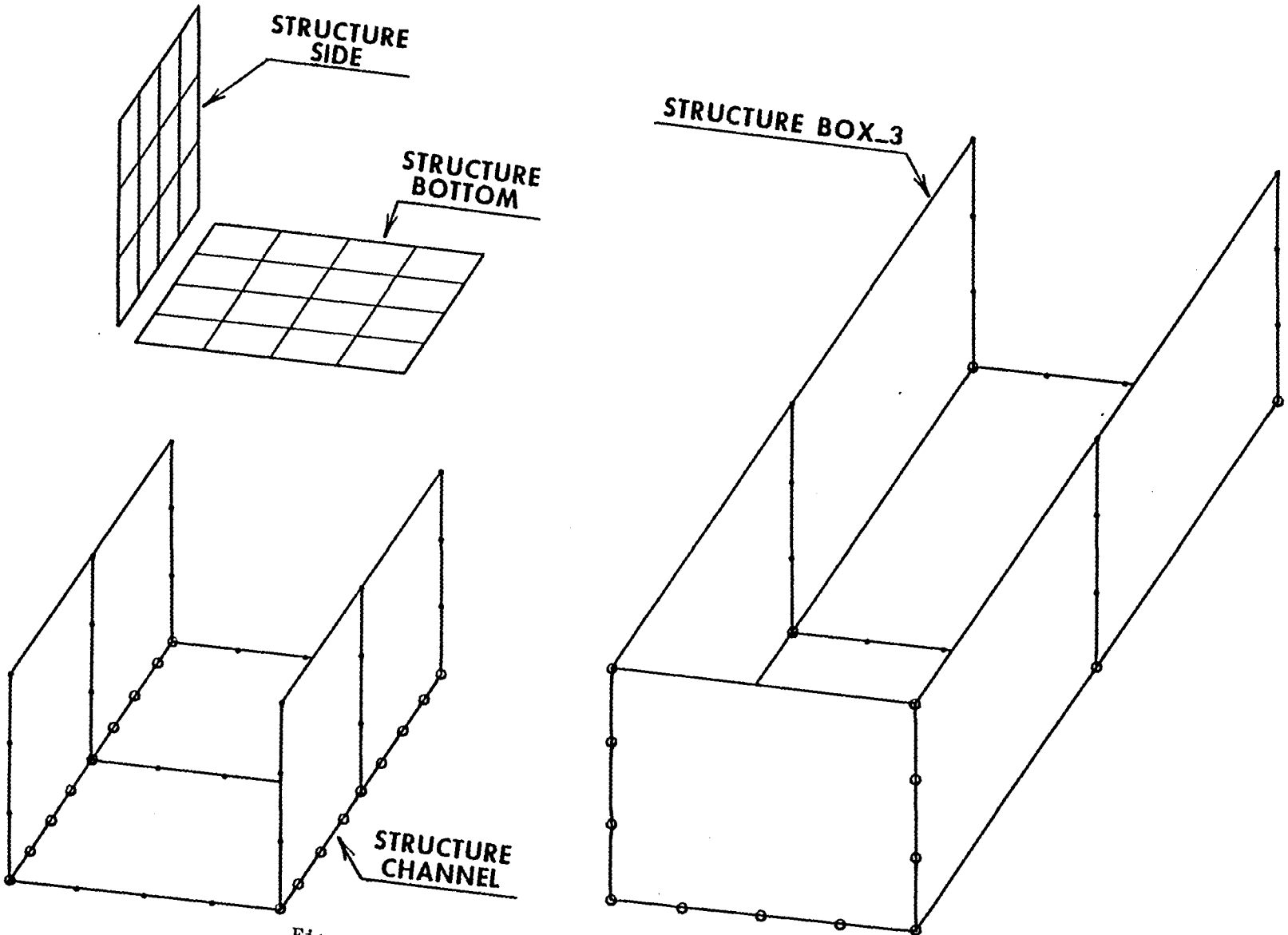


Figure 5.7. Finite Element Mesh for Structure BOX_3

data for this model are listed in Figure 5.8 and the structural hierarchy is presented in Figure 5.9. The first level of substructures is taken from the previous model, structures SIDE and BOTTOM, which are condensed into SIDE_CON and BOTT_CON, respectively. The second substructure level contains structure CHANNEL which consists of 4 condensed side panels and 2 condensed bottom panels. The condensed version of CHANNEL is CHAN_CON which contains the boundary nodes from CHANNEL and a selected number of retained normal DOF. The highest level structure, BOX_3, is assembled from two condensed channels and one condensed side panel. This structure contains 3 elements and 33 nodes (plus normal DOF).

One purpose of this example is to evaluate the performance of the fixed-interface method for the frequency analysis of a multilevel substructured model. The key parameter for study is the number of normal DOF retained in each of the reduced substructures. Table 5.1 lists the various combinations of normal DOF retained in each substructure. Structure BOX_2 was analyzed with four different combinations of normal DOF. These analyses are represented as 2A through 2D. Analyses were performed for structure BOX_3 using nine combinations of retained normal DOF. These analyses are identified as 3A through 3I.

Two types of comparisons are made for the analyses of this example. First, solution accuracy is evaluated. The errors in computing natural frequencies, mode shapes, and modal strains are examined for all substructured models. Approximate natural frequencies from the substructured models are compared directly against those for the baseline analysis. Mode shapes and modal strains are evaluated through

```
*RUN FINITE
C          MODEL 3E:  SUBSTRUCTURED VERSION OF THE CANTILEVER BOX
C                     MODEL.  SUBSTRUCTURES ARE REDUCED BY THE
C                     FIXED-INTERFACE METHOD.  THERE ARE 5 NORMAL
C                     DOF RETAINED IN EACH SUBSTRUCTURE.
C                     THIS MODEL USES TWO LEVELS OF SUBSTRUCTURING.
C                                 ==========
  ELEMENT WAFER
    TYPE RFSHELL CONSISTENT  E 30000. NU 0.3  THICKNESS 0.0625,
              SHORT OUTPUT   MASS_DENSITY 0.0007339  NOSPRINGS
    COORDINATES
      1   0.0   0.0
      2   0.75  0.0
      3   0.75  0.75
      4   0.0   0.75
C
  STRUCTURE SIDE
    NUMBER OF ELEMENTS 12  NODES 20
    ELEMENTS ALL TYPE WAFER  ROTATION BY COORDINATES
C
    COORDINATES
       1      0.0      2.25     0.0
       4      0.0      0.0      0.0
      17      0.0      2.25     3.0
      20      0.0      0.0      3.0
      GEN 1-4 IN X 1-17 BY 4 IN Y
C
    INCIDENCES
      GEN 3 IN X 4 IN Y AS 1-12 FROM 1 2 6 5 ADD 1 IN X 4 IN Y
C
    FREQUENCY ANALYSIS TYPE SUBSPACE
      PROPERTIES  NUM PAIRS 5 ITERATIONS 15  STURM CHECK
C
    CONSTRAINTS
       5-7, 9-11, 13-15  THETAX = 0.0
C
  STRUCTURE SIDE_CON
    NUMBER OF ELEMENTS 1 NODES 11
    ELEMENT 1 TYPE SIDE CONDENSED  RETAIN NORMAL 1-5
C
    INCIDENCES
      1  1-4, 8, 12, 16, 20, 19, 18, 17
C
  STRUCTURE BOTTOM
    NUMBER OF ELEMENTS 16 NODES 25
    ELEMENTS ALL TYPE WAFER  ROTATION BY COORDINATES
C
    COORDINATES
       1   0.0   0.0   0.0
       5   3.0   0.0   0.0
      21   0.0   0.0   3.0
      25   3.0   0.0   3.0
      GEN 1-5 IN X 1-21 BY 5 IN Y
    INCIDENCES
      GEN 4 IN X 4 IN Y AS 1-16 FROM 1 2 7 6 ADD 1 IN X 5 IN Y
C
    CONSTRAINTS
       7-9, 12-14, 17-19  THETAY  =  0.0
C
    FREQUENCY ANALYSIS TYPE SUBSPACE
      PROPS  NUM PAIRS 5 ITERATIONS 15 STURM CHECK
C
  STRUCTURE BOTT_CON
    NUMBER OF ELEMENTS 1 NODES 16
    ELEMENT 1 TYPE BOTTOM  CONDENSED  RETAIN NORMAL 1-5
C
    INCIDENCES
      1   1-5 10 15 20 25 24 23 22 21 16 11 6
C
  STRUCTURE CHANNEL
    NUMBER OF NODES 45 ELEMENTS 6
    ELEMENTS
      1-4 TYPE SIDE_CON  ROTATION SUPPRESSED
      5-6 TYPE BOTT_CON  ROTATION SUPPRESSED
C
    INCIDENCES
      GEN 1-2 FROM 1-4        12-14 21-18 BY -1   ADD 17
      GEN 3-4 FROM 11-8 BY -1 15-17 25-28         ADD 17
      GEN 5-6 FROM 4-8 15-17 25-21 BY -1 14 13 12  ADD 17
C
    CONSTRAINTS
       18-20, 26-28     THETAX = 0.0
       22-24            THETAY = 0.0
C
    FREQUENCY ANALYSIS TYPE SUBSPACE
      PROPS PAIRS 10 ITERATIONS 40 STURM CHECK
C
  STRUCTURE CHAN_CON
    NUMBER OF ELEMENTS 1 NODES 22
    ELEMENT 1 TYPE CHANNEL  CONDENSED  RETAIN NORMAL 1-5
C
    INCIDENCES
      1   1-11  35-45
C
  STRUCTURE BOX_3
    NUMBER OF ELEMENTS 3 NODES 33
    ELEMENTS
      1-2 TYPE CHAN_CON  ROTATION SUPPRESSED
        3 TYPE SIDE_CON  ROTATION Y  90.0
C
    INCIDENCES
      1   1-22
      2  12-33
      3  23-33
C
    CONSTRAINTS
       12-14 20-22     THETAX  =  0.0
       16-18           THETAY  =  0.0
       1-11  ALL  =  0.0
C
    FREQUENCY ANALYSIS TYPE SUBSPACE
      PROPS NUM PAIRS 10 ITERATIONS 30 STURM CHECK
C
    COMPUTE NATURAL FREQUENCIES
    OUTPUT  NATURAL FREQUENCIES   MODE SHAPES
    STOP
```

Figure 5.8. POL Definition of Structure BOX_3

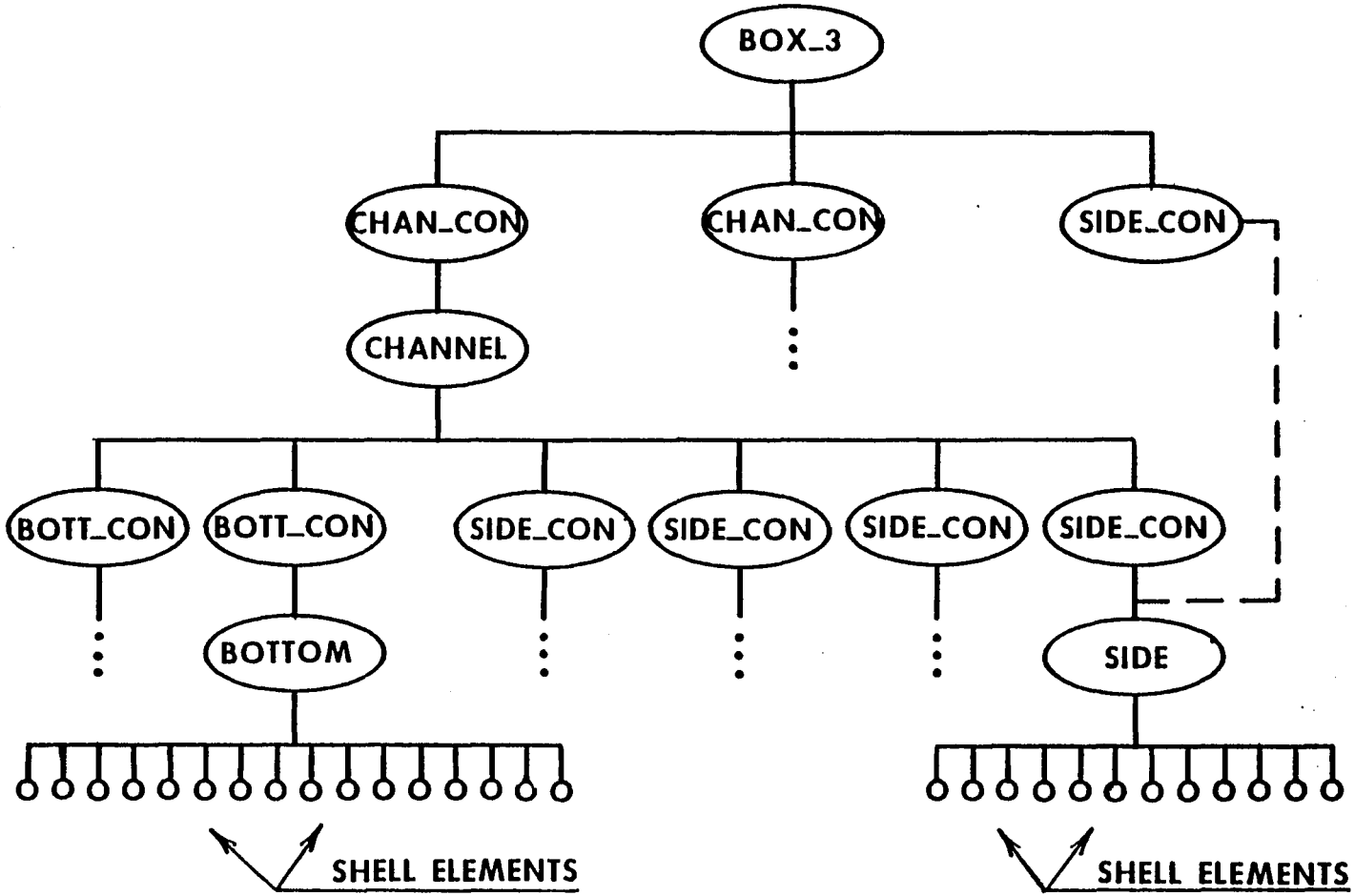


Figure 5.9. Hierarchy of Structure BOX_3

MODEL	SUBSTRUCTURE		
	SIDE_CON (30)[1]	BOTT_CON (45)[1]	CHAN_CON (129)[1,2]
2A	0	0	---
2B	5	5	---
2C	10	10	---
2D	15	15	---
3A	0	0	0
3B	0	0	5
3C	0	0	10
3D	5	5	0
3E	5	5	5
3F	5	5	10
3G	10	10	0
3H	10	10	5
3I	10	10	10

Notes

[1] Numbers is parenthesis indicate the number of interior nodal DOF in the parent substructure.

[2] Models 2A-2D do not contain substructure CHAN_CON.

Table 5.1 -- Number of Retained Normal DOF in BOX Models

a pair of error norms which represent the overall quality of these approximate vectors. The second comparison focuses on the costs of performing the analyses. Both CPU and paging requirements are examined. CPU requirements are measured by recording the amount of time used by the computer's central processor in solving the problem. Paging is measured as the number of page faults (or page replacements) performed by the POLO memory manager.

Table 5.2 lists the first 10 natural frequencies for the non-substructured model (BOX_1) and the corresponding errors in natural frequencies for the substructured models. Results for only 10 of the 13 substructured analyses are listed in the table. For models 2B, 2C, and 2D, computed frequencies for all 10 modes matched the baseline frequencies to 4 significant figures. Examination of Tables 5.1 and 5.2 reveals that the substructured frequencies converge to the baseline frequencies when at least 5 normal DOF are retained in each substructure. This condition exists for models 2B, 2C, 2D, 3E, 3F, 3H, and 3I. The maximum error in any of the 10 natural frequencies for these models is only 2.0% with a mean error of 0.8%.

The need to retain normal DOF in the highest level structure is demonstrated by the results for models 3A, 3D, and 3G. In these models Guyan reduction is applied to condense the second level substructure (CHANNEL). The results for these models are sufficiently poor to preclude their use in practical applications. The results for models 3D and 3G, which contain normal DOF in the first level substructures but not in the second, show no measurable improvement over results for model 3A, in which Guyan reduction was used at each substructure level. The

MODE	ω (rad/sec) BOX-1	PERCENT ERROR IN ω MEASURED AGAINST BOX-1 VALUES[1] SUBSTRUCTURED MODEL									
		2A	3A	3B	3C	3D	3E	3F	3G	3H	3I
1	50.92	0.5	5.5	1.7	1.7	5.5	1.2	1.2	5.5	1.2	1.2
2	55.78	0.9	8.6	1.7	1.7	8.6	0.8	0.8	8.6	0.8	0.8
3	81.61	1.5	9.2	2.2	2.2	9.2	0.8	0.7	9.2	0.8	0.7
4	89.31	1.9	12.8	2.9	2.9	12.8	0.9	0.9	12.8	0.9	0.9
5	98.25	0.7	49.8	2.5	2.5	49.1	2.0	2.0	49.1	2.0	2.0
6	110.3	2.5	68.5	3.0	3.0	68.5	0.5	0.5	68.5	0.5	0.5
7	138.8	3.9	69.6	3.0	3.0	54.6	-0.4	-0.6	54.6	-0.4	-0.7
8	153.2	3.5	56.2	2.9	2.9	56.1	0.2	-0.3	56.1	0.2	-0.3
9	198.0	15.0	30.1	17.7	17.2	30.1	0.6	0.3	30.1	0.6	0.3
10	214.1	8.4	68.8	11.5	10.4	67.3	0.6	0.4	67.3	0.6	0.3

[1] Models 2B, 2C, and 2D are exact to within 4 significant figures.

Table 5.2 -- Natural Frequencies for BOX Models

retention of normal modes in the lower level does not appear to influence the quality of results for the higher level substructures if the later are condensed by Guyan reduction. This effect is not unexpected in light of the procedure developed for selection of master and slave DOF (Section 2.3.2). The normal DOF in structures SIDE_CON and BOTT_CON of models 3D and 3G are grouped as slave DOF when assembled into structure CHANNEL. As such, their influence is eliminated from the model when Guyan reduction is applied to reduce structure CHANNEL into structure CHAN_CON.

Models 3B and 3C produce sizable errors in natural frequency, relative to models 3D, 3E, 3H, and 3I. This is due to the absence of normal DOF in substructures SIDE_CON and BOTT_CON in these models. The need for retained normal DOF at all levels of the structural hierarchy is clearly demonstrated in this example.

Model 2A, which employs Guyan reduction of all substructures, shows reasonable accuracy in natural frequencies. This is due to the greater number of nodes in the highest level structure compared to 3A, 3D, and 3G (79 versus 33) and to the more uniform distribution of those nodes (compare Figures 5.4 and 5.7).

The quality of a DOF reduction technique for dynamic analysis should not be evaluated solely on the basis of natural frequencies. The computed mode shapes and modal strains for the substructured box models are also examined in this example to assess the accuracy of the reduction. Results from the analysis of the non-substructured model (BOX_1) again provide a baseline for comparison. The results from the substructured models (2A-2D, 3A-3I) are taken as the approximate values for which error norms are calculated.

To obtain a meaningful comparison between results from the baseline and from the substructured models, the mode shapes for the substructured models are transformed to the geometric coordinates of the substructures at the lowest level of the hierarchy (see Section 2.3.1). A one-to-one correspondence then exists between terms of the baseline and of the approximate mode shapes.

Modal strains are computed for the individual finite elements using the mode shapes as displacement vectors. After strains for each element are computed, strains at the nodes are computed as the average of the contributions from all elements incident on a given node. Only nodes which join coplanar elements are considered. Nodes along the boundaries of the panels are not included in the comparison since the shell element is not expected to perform well at these locations [8]. The six strain components evaluated at the nodes are:

$$\begin{aligned} \epsilon_1 &= \partial u/\partial x, & \epsilon_4 &= \partial^2 w/\partial x^2 \\ \epsilon_2 &= \partial v/\partial y & \epsilon_5 &= \partial^2 w/\partial y^2 \\ \epsilon_3 &= \partial u/\partial y + \partial v/\partial x & \epsilon_6 &= \partial^2 w/\partial x \partial y \end{aligned} \qquad (5.1)$$

No changes are made in normalization of the mode shape vectors prior to performing the comparisons. As they are computed, the mode shapes are scaled to be orthonormal with respect to the mass matrix of the structure. For the non-substructured model (the baseline), the mass matrix may contain only geometric coordinates. For the substructured models, the structure mass contains both geometric and normal coordinates (a consequence of the substructure reduction procedure). This

apparent difference is not relevant since the mode shapes for the substructures are recovered completely to the lowest level of the hierarchy where all coordinates are geometric.

The quality of the approximate mode shapes and modal strains is evaluated through the computation of two error norms. The two norms, L_1 and L_2 [51], are defined by

$$L_1 = \frac{\frac{1}{n}\sum_{1}^{n} |d_i - \bar{d}_i|}{|\bar{d}_{max}|} \times 100\% \quad \text{and} \tag{5.2}$$

$$L_2 = \frac{\left(\frac{1}{n}\sum_{1}^{n} (d_i - \bar{d}_i)^2\right)^{0.5}}{|\bar{d}_{max}|} \times 100\% \tag{5.3}$$

in which: d_i is the i^{th} term in the approximate vector,

$\bar{d}_i$ is the i^{th} term in the baseline vector,

$\bar{d}_{max}$ is the largest term in the baseline vector, and

n is the number of terms in the baseline vector.

Table 5.3 lists the L_1 norms for mode shapes for all substructured models (2A-2D, 3A-3I). The values in this table exhibit the same trends established in Table 5.2 for the natural frequencies. Table 5.3 shows slightly larger error norms for modes 5 and 10 relative to the other modes. Apparently, an essential component of structure response for these modes is omitted from the models by truncation of the normal DOF. The models in which normal modes are retained at each level of the hierarchy (2B-2D, 3E, 3F, 3H, and 3I) predict mode shapes with the least

MODE	MODEL												
	2A	2B	2C	2D	3A	3B	3C	3D	3E	3F	3G	3H	3I
1	0.3	0.1	0.2	0.1	1.7	0.5	0.5	1.7	0.4	0.4	1.7	0.4	0.4
2	0.3	0.1	0.3	0.4	1.8	0.5	0.5	1.8	0.4	0.4	1.8	0.4	0.4
3	0.4	0.1	0.4	0.2	2.9	0.5	0.5	2.9	0.3	0.3	2.9	0.3	0.3
4	0.9	0.1	0.4	0.9	12.9	0.6	0.6	12.8	0.6	0.6	12.8	0.6	0.6
5	1.6	0.1	0.3	0.7	24.3	1.0	1.0	24.4	1.7	1.6	24.4	1.7	1.6
6	0.6	0.1	0.6	0.3	5.1	0.7	0.7	5.1	0.3	0.3	5.1	0.3	0.3
7	1.4	0.4	1.0	1.7	10.2	1.6	1.6	10.1	0.6	0.6	10.1	0.6	0.6
8	1.3	0.3	1.0	0.6	13.7	1.5	1.5	13.8	0.7	0.5	13.8	0.7	0.5
9	15.8	0.2	2.3	3.3	13.7	7.8	8.8	13.7	0.8	0.6	13.7	0.8	0.6
10	11.7	0.4	1.7	1.6	11.2	13.1	13.1	11.2	1.2	1.0	11.2	1.2	1.0

Table 5.3 -- L_1 Norm for Mode Shapes -- BOX Models

error. Some variability in L_1 is evident for models 2B-2D while the norms for the other four models are virtually identical to each other.

The L_2 norms for the same mode shape vectors are listed in Table 5.4. By design, the L_2 norm emphasizes regions of the approximation vector where the error function $(d_i - \bar{d}_i)$ attains its maximum value. Since the L_2 norms are 2-5 times larger than the associated L_1 norms, regions of "higher-than-average" error are indicated. However, the errors remain well within reasonable engineering accuracy for models in which natural frequency is well predicted.

The L_1 and L_2 error norms for the approximate modal strains are listed in Tables 5.5 and 5.6, respectively. The effects of numerical differentiation of the mode shapes to obtain the strains are clearly shown in these tables. While the trends established in the examination of mode shapes are repeated for modal strains, the magnitudes of the error norms are larger.

The effects of truncation of the normal DOF from the condensed substructures are well illustrated in this example structure. The natural frequencies are well predicted when normal DOF are retained in the reduced substructures. Computation of modal strains resulted in error norms that are higher than those for mode shapes. Within the individual modal strain vectors, the lowest values for the error function $(d_i - \bar{d})$ are obtained for strain components ϵ_1 and ϵ_2. As expected, error values increase for the remaining components of strain as the order of the numerical differentiation increases.

MODE	MODEL												
	2A	2B	2C	2D	3A	3B	3C	3D	3E	3F	3G	3H	3I
1	0.7	0.5	0.6	0.6	2.9	1.0	1.0	2.9	0.9	0.9	2.9	0.9	0.9
2	1.2	1.1	1.2	1.4	3.6	1.3	1.3	3.6	1.3	1.2	3.6	1.3	1.2
3	1.2	0.8	1.2	1.0	5.9	1.2	1.2	5.9	0.9	0.8	5.9	0.9	0.8
4	1.6	0.6	1.2	2.2	20.8	1.4	1.4	20.8	1.1	1.1	20.8	1.1	1.1
5	2.9	0.6	1.0	1.8	41.2	1.7	1.7	41.4	2.7	2.7	41.4	2.7	2.7
6	1.4	0.5	1.8	1.1	12.0	1.4	1.4	12.0	0.5	0.4	12.0	0.5	0.4
7	2.9	2.0	2.9	4.6	20.9	3.2	3.2	22.0	1.3	1.2	22.0	1.3	1.2
8	2.9	1.4	3.0	2.1	20.8	3.0	3.0	21.1	1.4	1.1	21.1	1.4	1.1
9	22.4	1.0	7.3	9.4	25.3	13.8	15.3	25.4	1.6	1.3	25.3	1.6	1.3
10	19.2	2.7	5.5	5.5	23.4	20.3	20.4	23.7	1.8	1.4	23.7	1.8	1.4

Table 5.4 -- L_2 Norm for Mode Shapes -- BOX Models

MODE	MODEL												
	2A	2B	2C	2D	3A	3B	3C	3D	3E	3F	3G	3H	3I
1	2.4	1.3	3.1	3.0	5.5	3.1	3.0	5.5	2.3	2.1	5.5	2.3	2.1
2	1.5	0.6	1.7	2.3	4.1	1.6	1.6	4.1	0.9	0.7	4.1	0.9	0.7
3	2.3	1.2	2.8	2.7	5.7	2.4	2.4	5.7	1.6	1.4	5.7	1.5	1.4
4	3.0	1.2	3.1	4.6	13.9	3.0	3.0	13.9	1.4	1.3	13.9	1.4	1.2
5	5.8	2.1	3.3	4.6	36.7	5.2	5.2	37.5	4.0	3.8	37.5	4.0	3.8
6	3.0	1.5	4.3	3.4	10.3	3.0	3.0	10.4	1.6	1.6	10.4	1.5	1.5
7	3.4	1.3	3.5	4.5	11.3	3.7	3.7	12.2	1.9	1.6	12.3	1.9	1.5
8	3.3	1.4	4.0	3.3	10.5	3.5	3.5	10.7	2.1	1.7	10.7	2.1	1.6
9	9.5	0.8	4.1	4.4	15.5	10.2	11.1	15.5	1.6	1.3	15.5	1.6	1.3
10	13.1	1.2	4.6	4.4	9.0	8.0	8.2	9.4	1.6	1.4	9.4	1.6	1.3

Table 5.5 -- L_1 Norm for Modal Strains -- BOX Models

MODE	MODEL												
	2A	2B	2C	2D	3A	3B	3C	3D	3E	3F	3G	3H	3I
1	7.2	4.4	8.1	8.0	13.0	7.7	7.7	13.1	5.4	5.1	13.1	5.4	5.0
2	4.0	2.2	4.5	6.1	9.7	4.2	4.2	9.7	2.6	2.4	9.7	2.6	2.4
3	7.1	4.0	7.2	7.3	14.5	7.2	7.2	14.5	4.4	4.2	14.5	4.4	4.2
4	7.3	3.4	7.7	11.8	26.5	7.7	7.7	26.5	3.7	3.6	26.5	3.6	3.6
5	13.2	5.5	8.1	12.3	90.1	11.4	11.5	91.6	8.1	7.7	91.6	8.0	7.6
6	8.7	5.0	11.2	8.6	30.4	8.8	8.8	30.5	4.9	4.9	30.5	4.8	4.9
7	8.5	3.9	8.5	11.1	25.5	8.8	8.9	29.9	4.4	3.6	29.8	4.4	3.6
8	9.0	4.6	4.7	8.0	21.7	9.0	9.0	22.2	5.3	4.3	22.0	5.3	4.3
9	18.6	2.3	10.9	10.6	36.2	21.2	23.2	36.2	3.4	2.8	36.2	3.8	3.4
10	25.9	4.4	11.9	11.8	21.6	17.5	17.9	25.0	3.7	3.3	25.1	3.8	3.4

Table 5.6 -- L_2 Norm for Modal Strains -- BOX Models

For the analyses discussed above, a convergence tolerance of 10^{-6} on eigenvalues was used in frequency analysis at all levels of the hierarchy. To check convergence, model 3E was re-analyzed with a tolerance of 10^{-10}. No improvement in frequencies, mode shapes, or modal strains was observed. This test verified that convergence of frequencies to a tolerance of 10^{-6} did not result in termination of the analysis before the mode shapes fully converged.

The computational effort for analysis of the substructured cantilever box models is summarized in Figure 5.10. The data are plotted against the CPU time and the number of page faults required for analysis of structure BOX_1. In all cases significant savings were realized in both CPU time and paging for the analysis of the substructured models. Also as expected, the multilevel substructured models, 3A - 3I, produced greater savings than did models 2A - 2D.

For all substructured analyses, the efficiency gained in paging exceeds that obtained for CPU time. This result is attributed to the smaller databases required for the substructured models. In general, only a small portion of the problem data can reside in memory at any one time. Since the number of pages in the working set (or dynamic pool) was held constant for all analyses performed in this example, proportionatly fewer page faults were needed to access data for the smaller models. Simply stated, for smaller models more of the database resides in the working set for longer intervals resulting in fewer page faults. In contrast, CPU performance is dominated by the number of computations required for eigensolution. Working set size has little influence on

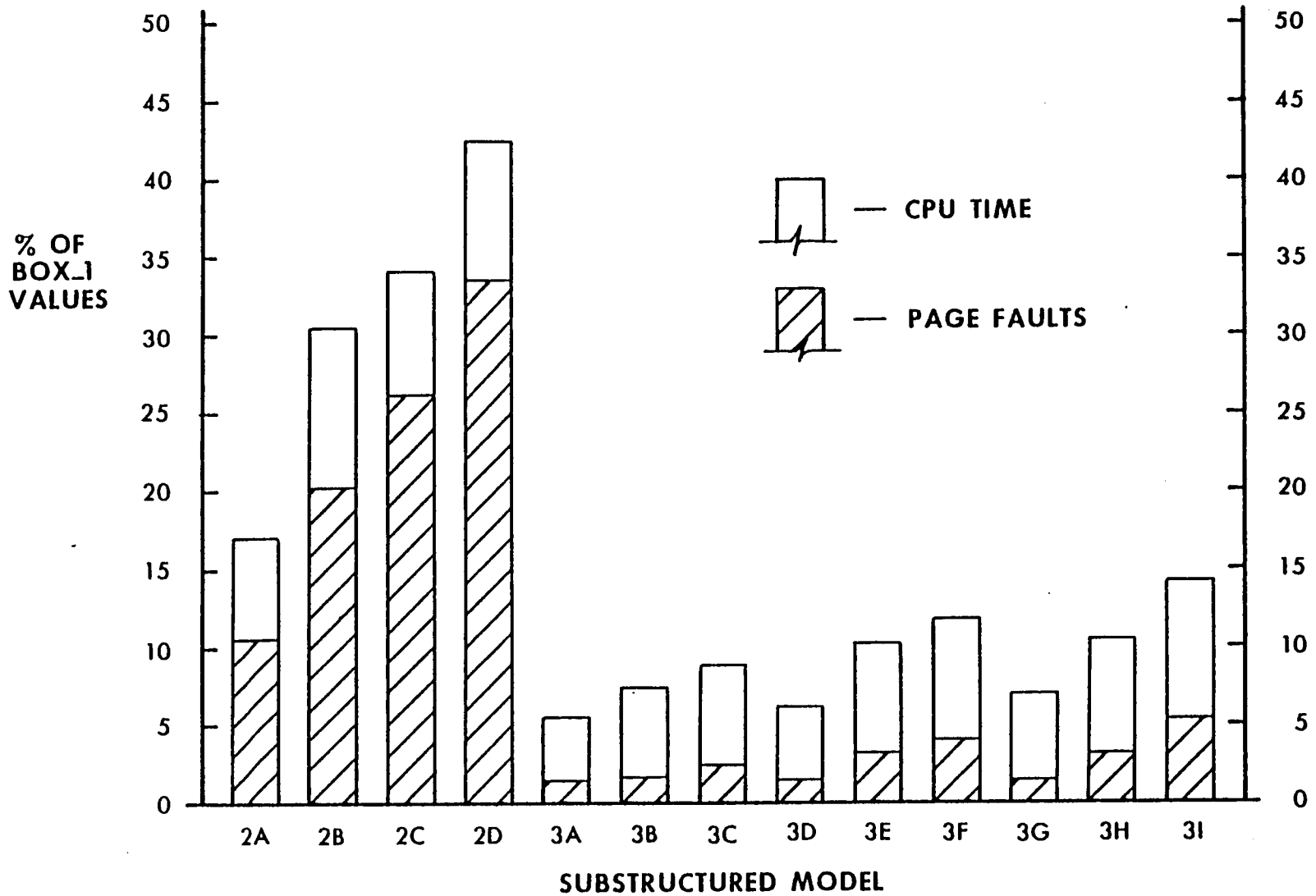


Figure 5.10. CPU and Paging Performance of BOX Models

the CPU time for such computationally intensive problems. Thus paging efficiency exceeds CPU efficiency in this example.

The accuracy and economy of the fixed-interface method for models using one level of substructuring has been previously noted [1, 10, 27]. Based on the results of this example problem, computational efficiency is further improved at no loss in solution accuracy when the fixed-interface method is applied to multilevel substructured models. The results for models 2B and 3E clearly demonstrate the advantage of multi-level substructuring. Computed frequencies, mode shapes, and modal strains are virtually identical but model 3E required only 33% of the CPU time and 16% of the page faults needed by model 2B. Compared to the baseline analysis, model 3E yielded savings of 90% for CPU time and 97% for paging. Similar reductions in computational effort are anticipated for other classes of structures.

5.3 Double Tetrahedron

The purpose of this example is to highlight the modeling techniques and computational efficiency that are provided by substructured modeling in dynamic analysis. Emphasis is placed on the unique modeling procedures to handle a structure's rigid-body modes, to restart the frequency analysis of the parent structure, and to increase the number of normal DOF of a previously assembled child structure. While still critical to the success of the analysis, solution accuracy is evaluated only on the basis of natural frequencies.

The example structure is a space truss built in the form of a double tetrahedron. The structure is modeled with simple three-dimensional truss elements. The outline of the structure and the

support conditions are illustrated in Figure 5.11. The nine line elements in the figure are actually identical joist-like members composed of 90 truss elements each. The geometry of one of these joists is illustrated in Figure 5.12. Each joist consists of 10 triangular transverse panels joined by longitudinal and diagonal truss elements. For clarity the diagonal elements are omitted from the figure. At each end of the joist are three additional truss elements that meet at a single node. These end nodes are used for connectivity to the remainder of the structure. Figure 5.13 shows the fully assembled structure. Diagonal truss elements are again omitted from the joist members for clarity. Since the truss elements contain only translational DOF at the nodes, the entire structural system contains 10 rigid-body modes: one rigid-body rotation for each joist about its own local x-axis and one rigid-body rotation of the entire structure about an axis through its ball-and-socket supports.

The baseline model for this structure, given code name C1, uses a consistent mass formulation and no condensation of the joist members. Figure 5.14 lists the input data that defines this model. Structure JOIST is defined only once and then used nine times with different orientations in structure TETRA. A lumped mass model, code named L1, is used as a companion to the baseline model. This second model is identical in all respects except mass formulation. This change is made by replacing the mass formulation key word "CONSISTENT" with the key word "LUMPED." The consistent mass and lumped mass analyses are examined separately since the natural frequencies for each are expected to differ slightly. The approximate models use both lumped and consistent mass formulations and varying degrees of condensation of the JOIST

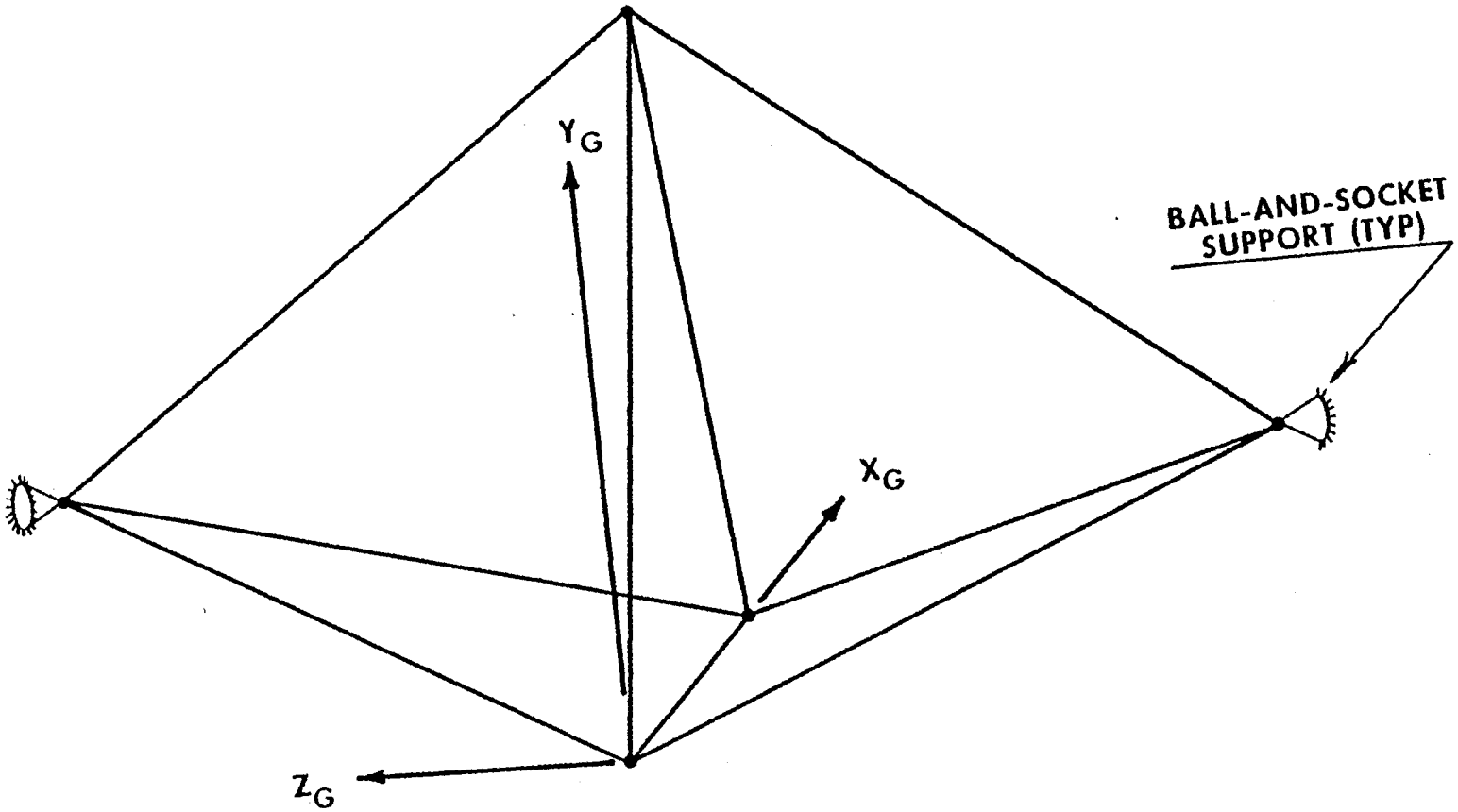


Figure 5.11. Double Tetrahedron Model

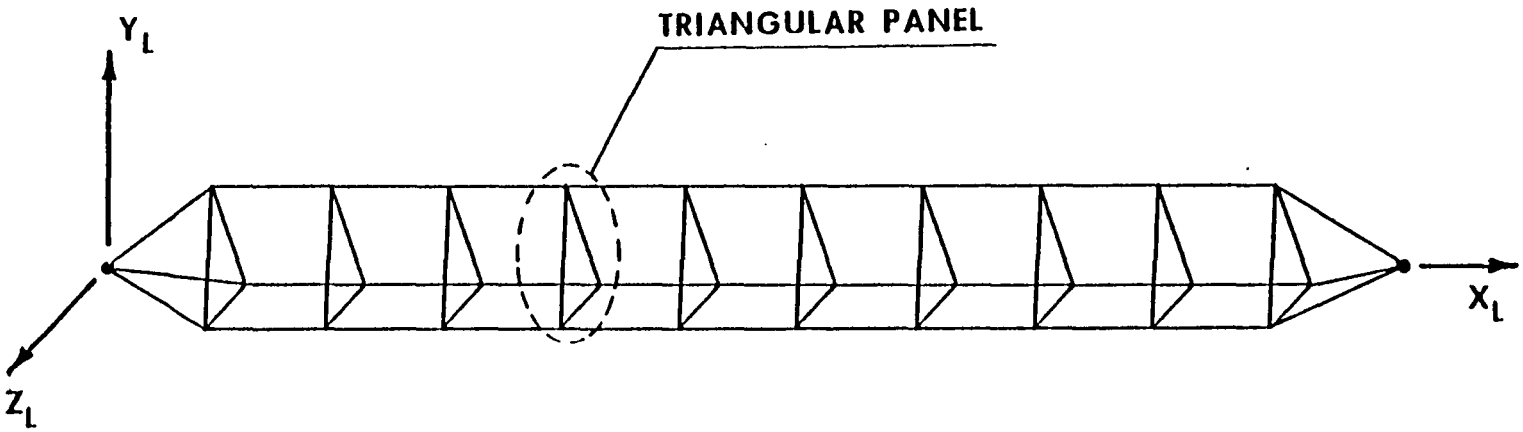


Figure 5.12. Finite Element Mesh for Structure JOIST

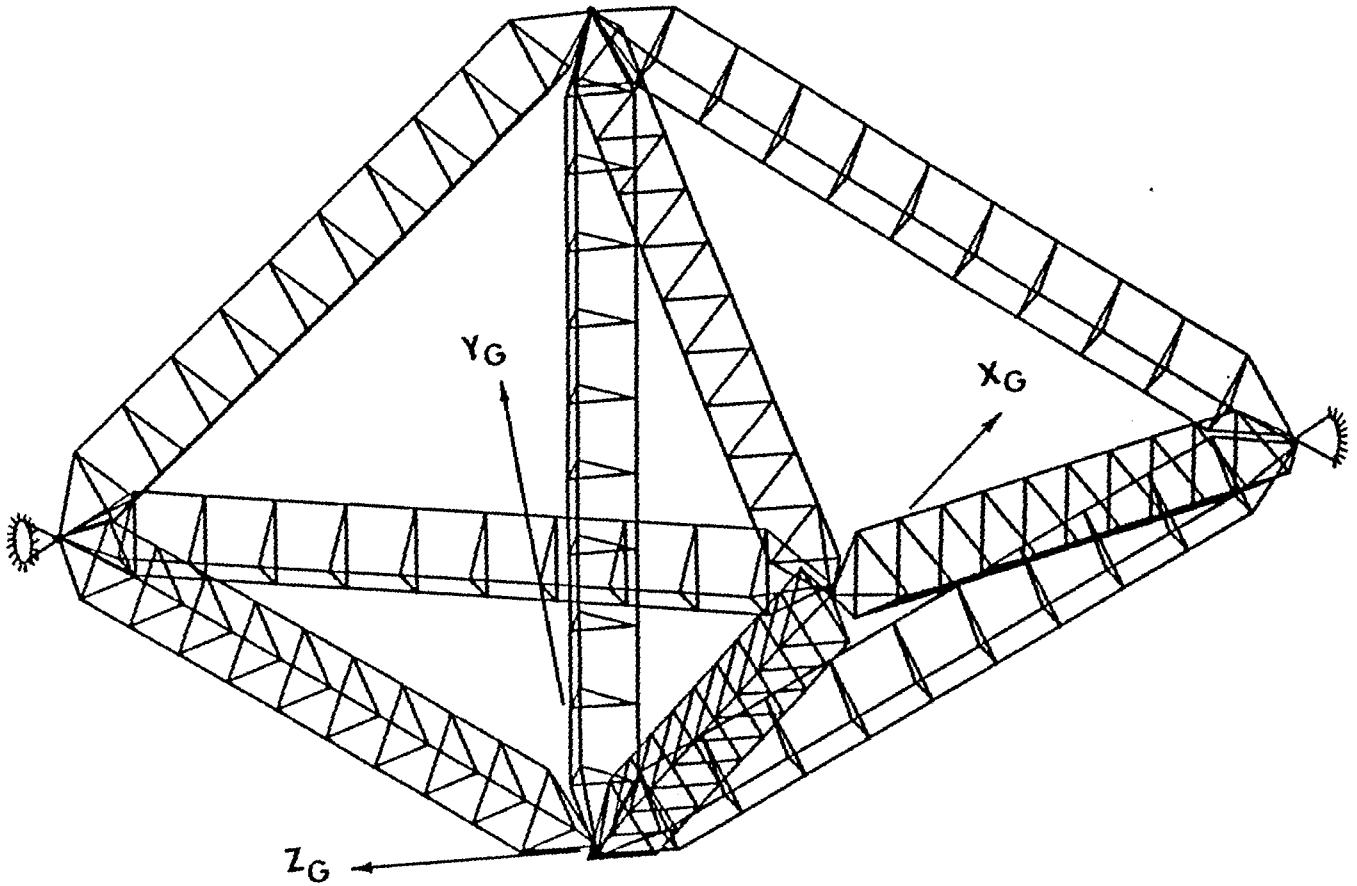


Figure 5.13. Finite Element Mesh for Structure TETRA

```
*RUN FINITE
C
C                SPACE TRUSS STRUCTURE USED TO DEMONSTRATE
C                RESTART OF SUBSPACE ITERATION, REANALYSIS OF
C                SUBSTRUCTURES, AND LUMPED AND CONSISTENT
C                MASS FORMULATIONS.
C
C                THE STRUCTURE USES SPACE TRUSS ELEMENTS TO BUILD A
C                LONG SLENDER JOIST SUBSTRUCTURE WHICH IS THEN USED
C                TO FORM THE NINE SIDES OF A DOUBLE TETRAHEDRON.
C
C                THIS IS THE NON-CONDENSED, CONSISTENT MASS VERSION.
C                         =============  ===============
C
  STRUCTURE JOIST
    NUMBER OF ELEMENTS 90 NODES 32
    ELEMENTS ALL TYPE SPACETRUSS CONSISTENT MASS_DENSITY 0.0007339,
                                          E 30000. AX 0.5
C
    COORDINATES
       1    0.0     0.0    0.0
       2   10.0     6.667  0.0
       3   10.0    -3.333  5.0
       4   10.0    -3.333 -5.0
      29  100.0     6.667  0.0
      30  100.0    -3.333  5.0
      31  100.0    -3.333 -5.0
      32  110.0     0.0    0.0
      GEN 2-29 BY 3  NOPRINT
      GEN 3-30 BY 3  NOPRINT
      GEN 4-31 BY 3  NOPRINT
C
    INCIDENCES
C
C              LONGITUDINAL CHORDS
C
      GEN 3 IN X 9 IN Y AS 1-27 FROM 2 5 ADD 1 IN X 3 IN Y
C
C              TRANSVERSE PANELS
C
      GEN 28-37 FROM 2 3 ADD 3
      GEN 38-47 FROM 3 4 ADD 3
      GEN 48-57 FROM 4 2 ADD 3
C
C              DIAGONALS
C
      GEN 58-66 FROM 2 6 ADD 3
      GEN 67-75 FROM 3 7 ADD 3
      GEN 76-84 FROM 4 5 ADD 3
C
C              PYRAMIDS AT ENDS
C
      GEN 85-87 FROM  1  2 ADD 0 1
      GEN 88-90 FROM 29 32 ADD 1 0
C
C
  STRUCTURE TETRA
    NUMBER OF NODES 275 ELEMENTS 9
    ELEMENTS   TYPE JOIST
      1  ROTATION  Y 121.482  Z  -16.102
      2  ROTATION  Y  58.518  Z  -16.102
      3  ROTATION  Y  90.0    Z   35.265
      4  ROTATION                                    SUPPRESSED
      5  ROTATION                          X  60.0
      6  ROTATION                          X 120.0
      7  ROTATION  Y  58.518  Z   16.102
      8  ROTATION  Y 121.482  Z   16.102
      9  ROTATION  Y  90.0    Z  -35.265
C
    INCIDENCES
      1   1-32
      2   1, 33-63
      3   1, 64-94
      4  32, 95-124,   63
      5  32, 125-154,  94
      6  63, 155-184,  94
      7  32, 185-215
      8  63, 216-245, 215
      9  94, 246-275, 215
C
    CONSTRAINTS
      1  215  ALL = 0.0
C
C              DEFINE THE FREQUENCY ANALYSIS, SHIFT FOR THE RIGID
C              BODY MODES OF THE STRUCTURE.
C
    FREQUENCY ANALYSIS TYPE SUBSPACE
      PROPS NUM PAIRS 15 ITERATIONS 20 STURM CHECK,
            RIGID BODY SHIFT -10.0
C
    COMPUTE FREQUENCIES
    OUTPUT  FREQUENCIES
    STOP
```

Figure 5.14. POL Definition of Double Tetrahedron

substructure. Figure 5.15 contains the input for a consistent mass model in which four normal DOF are retained in structure JOIST_CON, the condensed version of JOIST. Only the two end nodes of structure JOIST are retained in each approximate model. The condensed substructure, JOIST_CON, has just 6 geometric DOF. When a frequency analysis is required of a structure which contains rigid-body modes, the analyst specifies a small negative shift along with the other frequency analysis properties. This situation occurs both in the fixed-fixed frequency analysis of structure JOIST and in the free-free frequency analysis of structure TETRA. Subspace iteration is used for all frequency analyses performed in this example.

Table 5.7 provides a complete list of the code names, performance statistics, and modeling characteristics of each of the analyses performed for this example. The computed frequencies for the first five elastic modes from analyses C1 and L1 are listed in Table 5.8 along with the errors in computed frequencies for each of the approximate analyses. The structure's rigid-body modes were accurately evaluated in all the analyses and need not be listed.

The models in which structure JOIST is condensed by Guyan reduction, C2A and L2A, demonstrate the inadequacy of this approach for even a rough approximation of frequency response. Since all interior DOF are eliminated from structure JOIST_CON and no normal DOF are added, the rigid-body rotation and the internal elastic modes of each substructure are lost in the condensation process. The only elastic mode that structure JOIST_CON can exhibit is axial deformation. For analyses C2A and L2A structure TETRA has only one rigid-body mode: a rotation about its own support axis. The elastic modes predicted in these analyses do not

```
*RUN FINITE
C                DOUBLE-TETRAHEDRON MODEL
C                CONDENSED, CONSISTENT MASS VERSION
C                ==================================
C
C                THIS MODEL RETAINS 4 NORMAL DOF FROM SUBSTRUCTURE
C                JOIST IN THE CONDENSED SUBSTRUCTURE JOIST_CON.
C
  STRUCTURE JOIST
    NUMBER OF ELEMENTS 90 NODES 32
    ELEMENTS ALL TYPE SPACETRUSS CONSISTENT MASS_DENSITY 0.0007339,
                                          E 30000. AX 0.5
C
    COORDINATES
       1    0.0     0.0    0.0
       2   10.0     6.667  0.0
       3   10.0    -3.333  5.0
       4   10.0    -3.333 -5.0
      29  100.0     6.667  0.0
      30  100.0    -3.333  5.0
      31  100.0    -3.333 -5.0
      32  110.0     0.0    0.0
      GEN 2-29 BY 3
      GEN 3-30 BY 3
      GEN 4-31 BY 3
C
    INCIDENCES
C
C              LONGITUDINAL CHORDS
C
      GEN 3 IN X 9 IN Y AS 1-27 FROM 2 5 ADD 1 IN X 3 IN Y
C
C              TRANSVERSE PANELS
C
      GEN 28-37 FROM 2 3 ADD 3
      GEN 38-47 FROM 3 4 ADD 3
      GEN 48-57 FROM 4 2 ADD 3
C
C              DIAGONALS
C
      GEN 58-66 FROM 2 6 ADD 3
      GEN 67-75 FROM 3 7 ADD 3
      GEN 76-84 FROM 4 5 ADD 3
C
C              PYRAMIDS AT ENDS
C
      GEN 85-87 FROM  1  2 ADD 0 1
      GEN 88-90 FROM 29 32 ADD 1 0
C
    FREQUENCY ANALYSIS TYPE SUBSPACE
      PROPS NUM PAIRS 4 ITERATIONS 10 STURM CHECK,
            RIGID BODY SHIFT -10.0
C
  STRUCTURE JOIST_CON
    NUMBER OF NODES 2 ELEMENTS 1
    ELEMENT 1 TYPE JOIST CONDENSED   RETAIN NORMAL 1-4
C
    INCIDENCES
      1   1  32
C
  STRUCTURE TETRA
    NUMBER OF NODES 5 ELEMENTS 9
    ELEMENTS   TYPE JOIST_CON
      1  ROTATION  Y 121.482  Z  -16.102
      2  ROTATION  Y  58.518  Z  -16.102
      3  ROTATION  Y  90.0    Z   35.265
      4  ROTATION                          SUPPRESSED
      5  ROTATION                      X  60.0
      6  ROTATION                      X 120.0
      7  ROTATION  Y  58.518  Z   16.102
      8  ROTATION  Y 121.482  Z   16.102
      9  ROTATION  Y  90.0    Z  -35.265
C
    INCIDENCES
      1   1  2
      2   1  3
      3   1  4
      4   2  3
      5   2  4
      6   3  4
      7   2  5
      8   3  5
      9   4  5
C
    CONSTRAINTS
      1  5  ALL = 0.0
C
    FREQUENCY ANALYSIS TYPE SUBSPACE
      PROPS NUM PAIRS 15 ITERATIONS 100 STURM CHECK,
        RIGID BODY SHIFT -10.0 SUBSPACE SIZE 10
C
    COMPUTE FREQUENCIES
    OUTPUT  FREQUENCIES
    STOP
```

Figure 5.15. POL Definition of Condensed Double Tetrahedron

MODEL	CPU TIME	PAGE FAULTS	MASS FORMULATION	CONDENSATION COMMENTS
C1	1000	1000	CONSISTENT	DATUM MODEL: JOIST NOT CONDENSED
C2A	1.7	0.2	CONSISTENT	GUYAN REDUCTION (NO NORMAL DOF)
C2B	10.4	0.4	CONSISTENT	RETAIN NORMAL DOF 2-4 IN JOIST_CON
C2C	11.7	0.4	CONSISTENT	RETAIN NORMAL DOF 1-4 IN JOIST_CON
C2D	20.0	1.3	CONSISTENT	RESTART OF C2C: ADD NORMAL DOF 5-8
C2E	23.7	1.3	CONSISTENT	RETAIN NORMAL DOF 1-8 (VERIFY C2D)
L1	863.8	809.9	LUMPED	COMPANION TO DATUM: NO CONDENSATION
L2A	1.6	0.2	LUMPED	GUYAN REDUCTION (NO NORMAL DOF)
L2B	9.6	0.3	LUMPED	RETAIN NORMAL DOF 2-4 IN JOIST_CON
L2C	10.8	0.3	LUMPED	RETAIN NORMAL DOF 1-4 IN JOIST_CON

Table 5.7 -- Double Tetrahedron Model Characteristics

A. CONSISTENT MASS ANALYSIS

ELASTIC MODE	ω (rad/sec) MODEL C1	PERCENT ERROR MEASURED AGAINST C1 REDUCED MODEL				
		C2A	C2B	C2C	C2D	C2E
1	8.96	27.3	-1.06	0.60	0.56	0.56
2	9.20	27.2	-0.05	0.62	0.57	0.57
3	10.28	139.3	0.10	0.20	0.20	0.20
4	10.59	181.2	0.0	0.10	0.10	0.10
5	10.63	188.0	0.0	0.09	0.09	0.09

B. LUMPED MASS ANALYSIS

ELASTIC MODE	ω (rad/sec) MODEL L1	PERCENT ERROR MEASURED AGAINST L1 VALUES REDUCED MODEL		
		L2A	L2B	L2C
1	8.93	23.2	-2.54	0.08
2	9.16	23.9	-1.72	0.10
3	10.23	126.3	-0.10	0.0
4	10.54	165.0	-0.10	0.0
5	10.57	173.0	0.0	0.0

Table 5.8 -- Double Tetrahedron Natural Frequencies

correspond to the true behavior of the structure due to the absence of sufficient DOF in the final structure. In effect, Guyan reduction prevents the structure from vibrating at some of its lower natural modes. The application of Guyan reduction to this structural model clearly demonstrates its limited potential for accurate frequency analysis of substructured models.

Guyan reduction eliminates the rigid-body modes from the condensed substructures in analyses C2A and L2A. This characteristic is purposely used in analyses C2B and L2B to reduce the number of rigid-body modes in structure TETRA. For these analyses, the first 4 fixed-fixed normal modes are computed for structure JOIST. Mode 1 describes rigid-body rotation of the joist about its local x-axis. Modes 2-4 are elastic modes with non-zero frequencies. When JOIST_CON is defined, only normal modes 2-4 are retained through condensation. This procedure eliminates the rigid-body DOF from the substructure so that structure TETRA has only one rigid-body mode. Retention of normal modes 2-4 gives structure JOIST_CON elastic DOF which do not exist in the Guyan reduced models. The frequency results for these two analyses are close to those for the baseline but vary erratically. Normally, convergence to the baseline solution is monotonic from above. For C2B and L2B, some frequencies are underestimated, others are overestimated, and still others are virtually exact. Apparently, the rigid-body DOF neglected in the definition of JOIST_CON has an influence on the elastic modes of the structure and should be retained.

Analyses C2C and L2C include all four of the normal modes from structure JOIST in the condensation process, thus preserving the rigid-body mode of JOIST_CON. Input for C2C is listed in Figure 5.15. These

models provide a more consistent prediction of the natural frequencies for structure TETRA. For these two analyses, the lumped mass formulation shows slightly better convergence than does the consistent mass formulation but the data are insufficient to draw any general conclusions.

As a check on convergence of the consistent mass model, a partial reanalysis of C2C is performed to add the next 4 normal DOF from structure JOIST to structure JOIST_CON. The restart and reanalysis procedure is labeled analysis C2D. The reanalysis requires that the fixed-fixed frequency analysis of JOIST be restarted to compute modes 5-8. Substructure JOIST_CON is then re-defined to contain normal modes 1-8 in the reduction (modes 1-4 from the first analysis, modes 5-8 from the restart). The input commands for this analysis are shown in Figure 5.16. Three simple steps are involved in performing the analysis. First subspace iteration is restarted to compute the next 4 fixed-fixed eigenpairs of JOIST. The analyst defines the number of additional eigenpairs to compute and an initial shift value. Then, structure JOIST_CON is re-defined to contain the first 8 normal modes from structure JOIST. Finally, the frequency analysis for structure TETRA is requested. Characteristics of the structural model which do not change are not re-defined. For instance, the COORDINATES and INCIDENCES of structure JOIST are not repeated. Also, the orientation of each occurrence of JOIST_CON in TETRA remains unchanged during reanalysis so this data is not repeated. To the analyst, these model changes simply augment the description of the structural hierarchy. In fact, a major restructuring of the problem database takes place. However, this restructuring is transparent to the user.

```
*RUN FINITE FILES=20,21,22
C
C           DOUBLE TETRAHEDRON ANALYSIS  --  C2D
C           ====================================
C
C           RESTART ANALYSIS C2C TO ADD NORMAL DOF 5-8 TO
C           THE CONDENSED VERSION OF STRUCTURE JOIST.
C
C           THE FREQUENCY ANALYSIS OF STRUCTURE JOIST MUST BE
C           RESTARTED TO COMPUTE THE FIXED-FIXED FREQUENCIES
C           AND MODE SHAPES.
C
  ACCESS STRUCTURE JOIST   NONDESTRUCTIVE
C
    FREQUENCY ANALYSIS TYPE SUBSPACE
      PROPERTIES NUM PAIRS 4 ITERATIONS 20 STURM CHECK,
                 RIGID BODY SHIFT -10.0  MIN FREQ 0.13E04
C
C           DEFINE THE NEW LIST OF NORMAL DOF TO RETAIN IN
C           THE CONDENSED STRUCTURE.
C
  ACCESS STRUCTURE JOIST_CON   NONDESTRUCTIVE
C
    ELEMENT 1 TYPE JOIST CONDENSED   RETAIN NORMAL 1-8
C
C           RECOMPUTE FREQUENCIES FOR THE HIGHEST LEVEL
C           STRUCTURE.
C
  COMPUTE FREQUENCIES  FOR STRUCTURE TETRA
  OUTPUT  FREQUENCIES  FOR STRUCTURE TETRA
  STOP
```

Figure 5.16 -- POL Definition for Restart and Reanalysis

Analysis C2E is performed to verify the restart and reanalysis procedures of C2D. In analysis C2E, the first 8 fixed-fixed normal modes are computed for JOIST at the outset. All of these modes are then used in definition of JOIST_CON. This complete reanalysis procedure would be necessary to check convergence or to improve computed results had restart and partial reanalysis not been possible. In this example the computational costs between partial and complete reanalysis are almost the same. This is due to the relatively high overhead needed to support the restart and reanalysis procedure for such a small structural model. For larger models, analysis restart will be significantly more efficient than complete re-analysis of the model. Savings will be most evident when the costs for performing substructure reduction (fixed-fixed frequency analysis and the fixed-interface transformation) are a large portion of the cost for the entire structural analysis.

Performance statistics for all of the double-tetrahedron analyses are listed in Table 5.7. The CPU and paging requirements for the baseline analysis are assigned values of 1000 and results for the remaining 9 analyses are scaled accordingly. The condensation process provides a drastic reduction in computational expense compared to the non-condensed models. CPU and paging requirements are cut by up to two orders of magnitude in the approximate analyses. The potential for economical analysis of more practical structural systems is readily seen.

This example problem has demonstrated that the use of modal synthesis can produce orders-of-magnitude savings in computational effort while maintaining excellent accuracy. The analysis restart feature is an essential component of the software system. When there is doubt

about the quality of the reduced model, convergence testing can be conducted in an economical and convenient fashion. This flexibility encourages proper use of the advanced modeling and analysis techniques by both researchers and designers.

CHAPTER 6 -- SUMMARY AND CONCLUSIONS

6.1 Summary

Multilevel substructuring has been a popular technique for the economical analysis of complex structural models subjected to static loads. Modal synthesis is the collective name for techniques which extend the concept of substructuring to dynamic analysis. From this group of techniques, the fixed-interface method of Craig and Bampton was chosen as the focal point of study. Emphasis was placed on the implementation and performance of the method in POLO-FINITE, a general purpose software system which supports user-defined, multilevel substructured modeling.

The characteristics and analytical development of the fixed-interface method were discussed in detail. Advantages and disadvantages of the basic method were addressed, followed by a complete development of the procedure. The formulation was then extended to multilevel substructured modeling. Procedures for restart and reanalysis were also presented.

Software design and implementation was a major topic in this study. Application of the POLO executive for software development and run-time support was presented. POLO's two higher-level languages, DDL and HL, were reviewed. The function of each was illustrated through samples of the software developed for dynamic analysis. Integration of the hierarchical data structures, HL modules, and FORTRAN processing routines was also discussed.

The organization and control of the FINITE subsystems was reviewed for linear static and dynamic analysis. The POL that supports the new

modeling and analysis capabilities was discussed. Hypermatrix data structures and algorithms were presented as a basis for the computational procedures performed in FINITE. Control of the analysis procedures was reviewed for each of the new analysis functions implemented in this study. Implementation of frequency analysis procedures and of the fixed-interface method were presented in detail. The effects of hypermatrix data structures on the implementation were emphasized throughout. The procedure for restart and substructure reanalysis was outlined. The need for an effective data management executive to support this feature was demonstrated.

Two example structural systems were analyzed to demonstrate and evaluate the modeling and computational features of the FINITE system. These studies verified the accuracy and economy that is possible with multilevel substructured modeling. The generality of the implementation was shown to reduce both modeling effort and analysis costs while increasing flexibility.

6.2 Conclusions

The fixed-interface method provides a conceptually simple and reliable approach for the reduction of substructures for dynamic analysis. The method is applicable to multilevel substructured models and is compatible with flexible restart and reanalysis procedures. The fixed-interface method is a subset of several other modal synthesis techniques and thus provides an ideal choice for implementation in a general software system. While superior accuracy is sometimes possible with alternative synthesis methods, other considerations are equally important. Computational costs, user-interaction, and generality

(application to multilevel substructured models) must also be evaluated. These topics remain largely unstudied because of the lack of sophistication in other software systems used to evaluate modal synthesis techniques.

The generality of FEM software is equally dependent on the numerical algorithms that are chosen and on the software methodology used for implementation. General purpose software requires advanced techniques for data and computer resource management. Algorithmic languages do not support such tasks. The use of an executive system for development and run-time support becomes a necessity to modern analysis software. Restart and reanalysis are essential and natural features of dynamic analysis software that are generally neglected due to the complexity of the data management tasks. Implementation of this capability is dependent on the sophistication and versatility of the data manager within the executive.

The two example solutions clearly demonstrated the accuracy and efficiency of the software resulting from this study. For the first time, it has been demonstrated that fixed-interface reduction of multilevel substructured models can yield impressive savings in computational effort while maintaining good accuracy. Also, the unique restart and reanalysis procedures are simple to invoke so the analyst will be more willing to attempt convergence studies of the structural model.

The new modeling and computational components in POLO-FINITE establish the requisite tools for comprehensive studies in structural dynamics using substructured models. Extensive numerical testing is necessary to further evaluate the procedures for and consequences of substructure reduction.

The effects of the equation blocking precedure selected in Chapter 2 require additional study. Retained normal DOF are blocked as slave DOF when substructures containing reduced lower-level substructures as elements are themselves condensed. An alternative is to retain some normal coordinates as master DOF in higher level substructures. The result would be to lessen the detrimental effects of Guyan reduction (as illustrated in the cantilever box example, models 3A, 3D, and 3G) and to increase the size (order) of the higher level structure for subsequent analysis.

Implementation of standard dynamic analysis functions (transient analysis, shock spectrum response, etc.) in the POLO-FINITE system is now possible. The use of substructured modeling with time history integration is expected to yield significant reductions in both model development time and computational costs, paralleling those achieved in static analysis. A particularly promising area is the nonlinear analysis of substructured models in which the nonlinear response can be localized at the highest level of the hierarchy. Condensed, lower level substructures act as linear-elastic restraint on the nonlinear zone. As dynamic loading is applied, stiffness matrix updates are performed for only the nonlinear region. The linear substructures need not be re-condensed.

The application of time-dependent loads on reduced substructures presents a difficult implementation problem. Unlike static analysis, time-varying substructure loads cannot be simply condensed to the master DOF and carried forward in the hierarchy of the model. Special provisions must be made for time-history integration at the substructure level to fully evaluate these load effects.

REFERENCES

1. Bajan, R. L., Feng, C. C. and Jaszlics, I. J., "Vibration Analysis of Complex Structural Systems by Modal Substitution," Shock and Vibration Bulliten, vol. 39, no. 3, pp. 99-106 (1969)

2. Bamford, R., Wada, B. K., Garba, J. A. and Chisholm, J., "Dynamic Analysis of Large Structural Systems," Synthesis of Vibrating Systems, ASME Booklet, Nov. 1971, Library of Congress #76-179491

3. Bathe, K-J, and Wilson, E. L., "Large Eigenvalue Problems in Dynamic Analysis," Journal of Engineering Mechanics, ASCE, vol. 98, pp. 1471-1485 (1972)

4. Bathe, K-J, and Wilson, E. L., Numerical Methods in Finite Element Analysis, Prentice Hall, (1976)

5. Bathe, K-J, and Ramaswamy, S., "An Accelerated Subspace Iteration Method," Computer Methods in Applied Mechanics and Engineering, vol. 23, pp. 313-331 (1980)

6. Benfield, W. A. and Hruda, R. F., "Vibration Analysis of Structures by Component Mode Substitution," AIAA Journal, vol. 9, no. 7, pp. 1255-1261, (1971)

7. Braun, K. A., Dietrich, G., Frik, G., Johnsen, T. L., Straub, K., and Vallianos, G., "Some Hypermatrix Algorithms in Linear Algebra," Proceedings, Second International Symposium on Computing Methods in Applied Sciences and Engineering, Versailles (December, 1975)

8. Cook, R. D., Concepts and Applications of Finite Element Analysis, Second Edition, John Wiley and Sons (1981)

9. Corr, R. B. and Jennings, A., "A Simultaneous Iteration Algorithm for Symmetric Eigenvalue Problems," International Journal for Numerical Methods in Engineering, vol. 1, pp. 647-663 (1976)

10. Craig, R. R. and Bampton, M. C. C., "Coupling of Substructures for Dynamic Analysis," AIAA Journal, vol. 6, no. 7, pp. 1313-1319, (1968)

11. Craig, R. R. and Chang, C-J, "Free-Interface Methods of Substructure Coupling for Dynamic Analysis," AIAA Journal, vol. 14, no. 11, pp. 1633-1635, (1976)

12. Craig, R. R. and Chang, C-J, Substructure Coupling for Dynamic Analysis and Testing, NASA Contractors Report CR-2781, February (1977)

13. Craig, R. R., "Methods of Component Mode Synthesis," Shock and Vibration Digest, vol. 9, no. 11, pp. 3-10, (1977)

14. Craig, R. R. and Chang, C-J, "On the Use of Attachment Modes in Substructure Coupling for Dynamic Analysis," Proceedings of the 18th SDM Conference, San Diego, Cal. March 1977

15. Dodds, R. H. and Lopez, L. A., "Substructuring in Linear and Nonlinear Analysis," International Journal for Numerical Methods in Engineering, vol. 15, pp. 583-597 (1980)

16. Dodds, R. H. and Lopez, L. A., "Generalized Software for Nonlinear Analysis," International Journal for Advances in Engineering Software, vol. 2, no. 4, pp. 161-168 (1981)

17. Dodds, R. H., Rehak, D. R., and Lopez, L. A., "Development Methodologies for Scientific Software," Software - Practice and Experience, vol. 12, pp. 1085-1100 (1982)

18. Dodds, R. H., Rehak, D. R., and Lopez, L. A., "Software Virtual Machines for Development of Finite Element Systems," Proceedings of the 24th SDM Conference, Lake Tahoe, Nev., May, 1983

19. Fuchs, G. V., Roy, J. R., and Shrem, E., "Hypermatrix Solution of Large Sets of Symmetric Positive-Definite Linear Equations," Computer Methods in Applied Methanics and Engineering, vol. 1, pp. 197-216 (1972)

20. Furuike, T., "Computerized Multiple Level Substructured Analysis," Computers and Structures, vol. 2, pp. 695-712 (1972)

21. Gladwell, G. M. L., "Branch Mode Analysis of Vibrating Systems," Journal of Sound and Vibration, vol. 1, pp. 41-59, (1964)

22. Goldman, R. L., "Vibration Analysis by Dynamic Partitioning," AIAA Journal, vol. 7, no. 6, pp. 1152-1154, (1969)

23. Guyan, R. J., "Reduction of Stiffness and Mass Matrices," AIAA Journal, vol. 3, no. 2, p. 380, (1965)

24. Hale, A. L. and Meirovitch, L., "A General Substructure Synthesis Method for the Dynamic Simulation of Complex Structures," Journal of Sound and Vibration, vol. 69, no. 2, pp. 309-326 (1980)

25. Hale, A. L. and Meirovitch, L., "A Procedure for Improving Discrete Substructures Representation in Dynamic Synthesis," Proceedings of the 24th SDM Conference, Lake Tahoe, Nev., May 1983

26. Henshell, R. D. and Ong, J. H., "Automatic Masters for Eigenvalue Economization," Earthquake Engineering and Structural Dynamics, vol. 3, pp. 375-383 (1975)

27. Herting, D. N. "A General Purpose, Multi-Stage Component Modal Synthesis Method," Proceedings of the 20th SDM Conference, St. Louis, Mo., 1979

28. Hintz, R. M., "Analytical Methods in Component Modal Synthesis," AIAA Journal, vol. 13. no. 8, pp. 1007-1016, (1975)

29. Holze, G. H. and Boresi, A. P., "Free vibration Analysis Using Substructuring," Journal of the Structural Division, ASCE, vol. 101, pp. 2627-2639, (1975)

30. Hou, S-N, "Review of Modal Synthesis Techniques and a New Approach," Shock and Vibration Bulletin, vol. 4, no. 4, (1969)

31. Hurty, W. C., "Vibrations of Structural Systems by Component Mode Synthesis," Journal of the Engineering Mechanics Division, ASCE, vol. 86, no. 4, pp. 51-69, (1960)

32. Hurty, W. C., "Dynamic Analysis of Structural systems Using Component Modes," AIAA Journal, vol. 3, no. 4, pp. 678-685, (1965)

33. Hurty, W. C., Collins, J. D. and Hart, G. C., "Dynamic Analysis of Large Structures by Modal Synthesis Techniques," Computers and Structures, vol. 1, pp. 535-563, (1971)

34. Hurty, W. C., "Introduction to Modal Synthesis Techniques," Synthesis of vibrating Systems, ASME Booklet, Nov. 1971, Library of Congress #76-179491

35. Jennings, A. and Agar, T. J. A., "Progressive Simultaneous Inverse Iteration for Symmetric Linearized Eigenvalue Problems," Computers and Structures, vol. 14, no. 1-2, pp. 51-61 (1981)

36. Kidder, R. L., "Reduction of Structural Frequency Equations," AIAA Journal, vol. 11, no. 6, p. 892, (1973)

37. Kubomura, K., "A Theory of Substructure Modal Synthesis," Journal of Applied Mechanics, vol. 49, pp. 903-909, (1982)

38. Kuhar, E. J. and Stahle, C. V., "Dynamic Transformation Method for Modal Synthesis," AIAA Journal, vol. 12, no. 5, pp. 672-678, (1974)

39. Leung, Y. T., "An Accurate Method of Dynamic Condensation in Structural Analysis," International Journal for Numerical Methods in Engineering, vol. 12, pp. 1705-1715, (1978)

40. Leung, Y. T., "An Accurate Method of Dynamic Substructuring with Simplified Computation," International Journal for Numerical Methods in Engineering, vol. 14, pp. 1241-1256 (1979)

41. Lopez, L. A., "POLO - Problem Oriented Language Organizer," Computers and Structures, vol. 2, pp. 555-572, (1972)

42. Lopez, L. A., "FILES: Automated Engineering Data Management System," Journal of the Structural Division, ASCE, vol. 101, no. ST4, pp. 661-676 (1975)

43. Lopez, L. A., "FINITE: An Approach to Structural Mechanics Systems," International Journal for Numerical Methods in Engineering, vol. 11, no. 5, pp. 851-866 (1977)

44. MacNeal, R. H., "A Hybrid Method of Component Mode Synthesis," Computers and Structures, vol. 1, pp. 581-601 (1971)

45. Meirovitch, L. and Hale, A. L., "Synthesis and Dynamic Characteristics of Large Structures with Rotating Substructures," Dynamics of Multibody Systems, Symposium held in Munich, West Germany, Aug. 29 - Sept. 3, 1977, pp. 231-244

46. Meirovitch, L. and Hale, A. L., "A General Dynamic Synthesis for Structures with Discrete Substructures," Proceedings of the 21st SDM Conference, Seattle, Wash. May, 1980

47. Meirovitch, L. and Hale, A. L., "On the Substructure Synthesis Method," AIAA Journal, vol. 19, no. 7, pp. 940-947, (1981)

48. Miller, C. A., "Dynamic Reduction of Structural Models," Journal of the Structural Division, ASCE, vol. 106, pp. 2097-2108, (1980)

49. Morosow, G. and Abbot, P., "Mode Selection," Synthesis of Vibrating Systems, ASME Booklet, Nov. 1971, Library of Congress #76-179491

50. Przemieniecki, J. S., "Matrix Structural Analysis of Substructures," AIAA Journal, vol. 1, no. 1, pp. 138-147 (1963)

51. Rice, J. R., The Approximation of Functions, Addison-Wesley (1964)

52. Rubin, S., "Improved Component Mode Representation for Structural Dynamic Analysis," AIAA Journal, vol. 13, no. 8, pp. 995-1006,(1975)

53. Schmidt, R. J. and Dodds, R. H., Theoretical and Software Considerations for Nonlinear Dynamic Analysis, SM Report No. 8, Feb. (1983) University of Kansas, Center for Research, Lawrence, Kansas

54. Shah, V. N. and Raymund, M., "Analytical Selection of Masters for the Reduced Eigenvalue Problem," International Journal for Numerical Methods in Engineering, vol. 18, pp. 89-98 (1982)

55. Von Fuchs, G, Roy, J. R., and Schrem, E., "Hypermatrix Solution of Large Sets of Symmetric Positive-Definite Linear Equations," Computer Methods in Applied Mechanics and Engineering, vol. 1, pp. 197-216 (1972)

56. Williams, F. W., "Comparison of Sparse Matrix and Substruture Methods," International Journal for Numerical Methods in Engineering, vol. 5, pp. 383-394 (1973)

57. Wilson, E. L., "The Static Condensation Algorithm," International Journal for Numerical Methods in Engineering, vol. 8, no. 1, pp. 198-203 (1974)

58. Wilson, E. L. and Itoh, T., "An Eigensolution Strategy for Large Systems," Computers and Structures, vol. 16, no. 1-4, pp. 259-265 (1983)

59. Wright, G. C. and Miles, G. A., "An Economical Method for Determining the Smallest Eigenvalues of Large Linear Systems," International Journal for Numerical Methods in Engineering, vol. 3, pp. 25-33, (1971)

APPENDIX A -- USER INTERFACE AND INPUT DESIGN

A.1 General

The most popular approach to user communication with structural analysis software is the problem oriented language (POL). Virtually all successful software systems use the POL approach, either by initial design or by the use of pre-processors to translate POL input into fixed-format, card images. The POL approach provides the user with greater flexibility by placing him in control of the input process rather than forcing him to conform to rigid formats and input sequences. The self-documenting nature of the input reduces the need for reference to manuals and provides a concise description of the structural model for other analysts. The POL is essential for interactive processing in which error recovery is often necessary.

The philosophy established during the development of FINITE was to maintain as much independence as possible among the various components of a complete structural model. These components include nonlinear material models specification, geometric definition of the structures, parameters controlling nonlinear solution algorithms, and requests for computation and output. The primary reasons for choosing this approach are to provide maximum flexibility in using condensed substructures as elements in the higher level structures and to minimize the effect of changes in the structural model throughout the analysis/design sequence.

Wherever possible, this philosophy is maintained in the extension to dynamic analysis. One area does exist in which dynamic solution parameters must be tied directly to the geometric definition of a substructure. This is the frequency analysis of a substructure that is

to be condensed by modal synthesis. Since economical frequency analysis depends upon the type of structure, the number of eigenpairs required, and the solution method, it is not appropriate to select just one solution algorighm for all substructures in a complex model. Various substructures will have differing characteristics and may require an unequal number of retained normal modes for condensation. It is also possible that one substructure could be condensed two or more times in differing ways, with varying geometric and generalized DOF, for use in separate, higher level structures. Thus, it is necessary to tie the selection of the eigenproblem solution method to the structure definition.

The capabilities selected for general purpose dynamic analysis, along with the various options and parameters that control the solution, must be defined accurately and unambiguously by the POL. Section A.2 presents an explanation of the capabilities to be incorporated into POLO-FINITE. Section A.3 lists the syntax of the commands for dynamics and examples of their use. As stated earlier, this appendix describes the POL for a complete set of analysis capabilities, including those that have not been implemented as a part of this study. Portions of the POL which have not been implemented are indicated by an "*" in the section headings.

A.2 Description of the POL

A.2.1 Structure and Element Mass

The mass of a structure can be divided into two parts: primary and secondary. Primary mass is the mass of the load-carrying components (elements) of the structure. Its definition is easily added to the specification of an element through two new element properties. The first defines the type of mass formulation: LUMPED or CONSISTENT. The second is the MASS_DENSITY of the material of which the element is composed. The element mass matrix can then be formed using existing element shape functions. The FINITE system accepts up to thirty DOF at each node of an element. These include the translational DOF: U, V, and W, and their first and second derivatives: UX, VX, WX, UY, etc. Depending upon the particular element formulation, it is possible for mass to be assigned to any or all of these DOF.

Secondary mass is the mass of non-load-carrying components, such as concentrated and distributed live-loads, that are supported by the structure. Secondary mass is defined in a manner similar to the definition of gravity loads. The secondary mass is resolved into equivalent nodal mass via the appropriate element load shape functions. The result will always be a lumped mass matrix which is added to the primary mass of the structure. As with primary mass, secondary mass may be associated with any of the thirty nodal DOF.

There are three types of secondary mass: nodal, element, and pattern. Nodal mass is mass that is concentrated at a structure node. Element mass is concentrated or distributed on the surface of an element. Pattern mass enables the defintion of secondary mass in terms of a previously defined loading condition, usually gravity loading. The

user must specify only the name of the loading condition to be used as the pattern and a value for the acceleration of gravity to support the appropriate conversion from force to mass.

The commands for computation (assembly) and output of the mass matrix for a structure or stand-alone element follow directly from those for the stiffness matrix.

A.2.2 Structure Damping - *

Damping is typically defined only for the highest level structure, not for individual finite elements or substructures. Two methods are available for defining structural damping: modal and Rayleigh. Definition of modal damping requires input of the modal damping ratio for each vibration mode under consideration. Modal damping is applicable only to transient analysis by mode superposition. Rayleigh damping involves the definition of two damping ratios at two selected frequencies; the frequencies need not be eigenvalues of the structure. Rayleigh damping is applicable to transient analysis by either mode superposition or time-history integration. Use of Rayleigh damping requires that a frequency analysis be performed in order to compute the modal damping ratios for mode superposition or to explicitly form the damping matrix for time-history integration.

Depending upon the method used to define damping, either the damping matrix or modal ratios can be output for the structure.

A.2.3 Frequency Analysis

As previously mentioned, the parameters controlling the frequency analysis (computation of natural frequencies and mode shapes) must be

defined individually for each structure for which the analysis is to be performed. No default analysis method is adopted. The syntax for specification of the solution method is similar to that for a nonlinear material. The TYPE of solution procedure is identified followed by a listing of the PROPERTIES which control the procedure. Solution method properties can be changed via analysis restart. If a substructure is to be condensed by Guyan reduction, no frequency analysis specification is required.

The request for computation may be made explicitly by the analyst or the analysis may be invoked automatically by the FINITE processors. Standard output included natural frequencies and mode shapes. Recovery of mode shapes for condensed lower level substructures is performed when an output request is encountered to print those quantities. Substructures to be recovered are specified by appending a list of sub-element numbers to the name of the structure.

Prior to a transient analysis by mode superposition, the user may examine the modal content of a particular dynamic loading condition. A special output request facilitates selection of the modes that participate in the dynamic response. After a frequency analysis the analyst may request output of MODAL LOADS for the loading condition. The frequency content of the loading can then be examined and the appropriate modes selected for superposition.

As a tool for evaluation of the quality of the results in a modal synthesis analysis, MODAL STRAINS may be computed and output to the analyst. MODAL STRAINS are the element strains which result when a selected vibration mode shape is used as a displacement vector. Output

of MODAL STRAINS must be preceded by a frequency analysis of the structure.

A.2.4 Substructure Reduction

The procedure to request reduction of a substructure for dynamic analysis parallels that for static condensation. The reduction method is defined at the intermediate substructure level; i.e., the substructure with only one element of type CONDENSED. Guyan reduction is the default method. The fixed-interface method is invoked by specifying which substructure normal modes to retain. The modes specified must be within the range computed in the frequency analysis of the lower-level substructure which is being condensed. The retained modes need not be consecutively numbered. As an alternative to using substructure normal modes, user-supplied mode shapes can be used in the synthesis process. These modes could be derived from an experimental analysis or some other source, such as low-order polynomials. Input data describing these modes must be included with the definition of the structure to be condensed.

Reduction can be explicitly invoked with a COMPUTE STIFFNESS... or COMPUTE MASS... command for the intermediate level substructure. Reduction is performed automatically when required to satisfy a request for a higher-level structure.

A.2.5 Initial Conditions - *

Initial conditions can be defined for a structure prior to transient analysis. They define a starting solution, in terms of displacements and velocities, for the unconstrained physical DOF at time

t = 0. For all other times the displacements and velocities from the previous time step are used in the integration.

The analyst may specify initial conditions in one of two ways. First, he may define numerical values for each DOF with non-zero displacement or velocity. The default initial conditions are zero displacement and velocity for all unconstrained DOF. The second method uses the static equilibrium configuration from a previous linear or nonlinear analysis. This method allows the structure to be released from some deflected initial shape with zero initial velocity. A dynamic loading may then be applied as the transient response is evaluated.

A.2.6 Dynamic Loading - *

The dynamic loading function, P(x,y,z,t), is defined such that it has a spatially-varying component, F(x,y,z), and a time-varying component, G(t):

$$P(x,y,z,t) = F(x,y,z) * G(t). \qquad (A.1)$$

Simply stated, the pattern of the load is fixed and its magnitude changes with time.

The load pattern, F(x,y,z), can be described as either actual forces applied to the structure or as support accelerations. The former can best be defined as a static linear loading condition, while the latter requires an additional loading type: NODAL ACCELERATIONS. No special provisions are necessary for input of out-of-phase support accelerations. They can be recoginzed and handled automatically.

The time-varying component of the loading function, G(t), is defined along with other loading data in a dynamic loading condition. The G(t) vs. t relation may be harmonic, impulsive, or general. the

dynamic loading condition must also include the loading pattern, F(x,y,z), which is to be used. More than one static linear loading condition can be combined to form the complete pattern of the dynamic load. Other necessary input includes the values of time t at which displacements are to be computed (thus defining the integration step size) and values of time t at which computed results are to be retained in the data base. This last item is important because a transient analysis of any significant duration could result in more data than could be effectively stored. Also, it is likely that computed results would be required at only a few of the many time steps for which displacement are computed.

A.2.7 Transient Analysis - *

Transient analysis yields the displacement and velocity response of the structure when it is subjected to time-varying loading or support accelerations. Two approaches are available for performing transient analysis: mode superposition and time-history integration. Mode superposition requires that a frequency analysis be performed so the equations of motion can be uncoupled. This implies that an appropriate frequency analysis must be selected prior to requesting the transient analysis. The resulting set of independent equations is easily solved using one of the Lagrange interpolation formulae. Time-history integration is performed by any one of a number of explicit, implicit, or hybrid operators. Specification of the transient analysis method is similar to that for frequency analysis: the TYPE of method is defined followed by the PROPERTIES list.

The request for computation includes the structure to be analysed, the dynamic loading condition, time steps, and initial conditions. Results available for output include displacements, velocities, strains, and stresses.

A.2.8 Shock Spectrum Analysis - *

The analysis of shock spectrum response is currently restricted to linear structures. The shock spectrum is input by defining the functional relationship between a spatial coordinate and a time coordinate. The spatial coordinate can be chosen as displacement, velocity, or acceleration, while the time coordinate can be either period or frequency. The user inputs discrete points from the spectrum and the remainder of the curve is constructed by linear interpolation in four-way logarithmic coordinates. The direction of application of the shock is defined using direction cosines for the translational DOF (U, V, and W for 3-D structures). The nodes at which the shock is applied are also defined.

Prior to computing the spectral response, a frequency analysis of the structure must be performed. Spectral response quantities are computed only after the corresponding output request has been made. Results available for output include spectral displacements, spectral velocities, spectral strains, and spectral stresses. These quantities can be output on a mode-by-mode basis or in some combined form. Methods used to combine the modal quantities include SRSS (square root of the sum of the squares) and PEAK_SRSS (peak response mode plus SRSS of the remaining modes). PEAK_SRSS is also known as the Naval sum. As a measure of the portion of the total mass responding to the shock in each mode, the modal PARTICIPATION_FACTORS can also be output.

A.3 POL Syntax and Examples

A.3.1 Syntax Conventions

The following is a description of the conventions used in this section to illustrate the FINITE command syntax.

A descriptor is used to identify the position and class of a data item in a particular FINITE command line. The descriptor is delimited by the characters "< >." The command

NUMBER OF NODES <integer>

implies that the word NODES is to be followed by an integer. As appropriate example is:

NUMBER OF NODES 100

The following are definitions of the descriptors used within the POL:

<integer> -- a series of digits optionally preceded by a plus or minus sign. Examples are 121, +300, -8 .

<real> -- a representation of a floating point number in either decimal or exponential form. Real numbers must contain a decimal point and may be signed. Examples are 1.0, -3.5, 5.2E-08 .

<number> -- either an integer or a real number may be input. The data item is converted to a real number.

<integer list> -- a sequence of integers. The sequence may be listed explicitly or defined over a range of integers with a constant increment. The default increment is 1 . Examples are: 1, 2, 4, 5, 8, 11; 1-10; 2-20 BY 2 .

<real list> -- a sequence of real numbers. Real lists have the same form as integer lists except that there is no default increment. Examples are: 1.0, 1.5, 2.0, 3.0; 0.0-2.5 BY 0.25 .

<number list> -- either an integer list or a real list is input. The data is converted to real.

<label> -- a series of letters and digits beginnings with a letter. Labels are used as names for identifying various entities. Examples are: PLANEFRAME, DEADLOAD_10 .

<string> -- any text enclosed within single or double quotes. An example is: "THIS IS A STRING" .

In some instances a description of the physical meaning of the data item is added to the class in the syntax of a descriptor. This is helpful in clarifying the use of the data item. For example a command of the form

STRUCTURE <structure name:label>

implies that the data item following the word STRUCTURE is a label defining the name of the structure.

It is not always necessary to completely spell out every word on a command line in order to have the command correctly translated. Many words can be abbreviated and these are identified in the command syntax by underlining. The underlined portions of words identify the minimum input necessary for proper command translation. Descriptors are not underlined but are replaced by an item of the specified class when applicable. If the command syntax has the form:

NUMBER OF NODES <integer>

the following is acceptable as input:

NUM OF NODE 10

When only one word from a group of words may be selected as input, the choices are listed one above the other and enclosed in braces, "{ }" . The command syntax

$$\text{COMPUTE} \left\{ \begin{array}{l} \text{STIFFNESS} \\ \text{DISPLACEMENTS} \end{array} \right\}$$

implies that any of the following commands are acceptable:

COMPUTE STIFF

COMPUTE DISPLACEMENTS

COMPUTE DISPL

When an entire word or phrase in the command is optional, it is enclosed within parentheses. The command with the syntax

NUMBER (OF) NODES <integer>

can be issued as

NUM NODES 100

When more than one word from a group of words may be selected, the group is enclosed in brackets, "[ ]" . The command

OUTPUT [DISPLACEMENTS
STRAINS
STRESSES]

implies that the user may request

OUTPUT DISPL STRAINS

Brackets and braces are combined to allow more flexibility in designing commands. The command syntax

<integer> [{ X / Y / Z } <number>]

implies that the user may enter data of the form:

1 X 0.0 Y 0.0 Z 5.0

2 X 1.0 Z 5.0

Continuation of an input line onto a second physical line is accomplished by placing a comma at the end of the line to be continued.

Comments may be placed in the data by placing a "C" in column 1 and a blank in column 2 of the comment line.

One method for line termination is to place dollar-sign "$" on the line. All entries on the line following the "$" are ignored by the translator and may be used for comments.

A.3.2 Syntax and Examples

A.3.2.1 Specification of Mass

Example of the command to specify primary mass:

```
ELEMENT 1 TYPE CSTRIANGLE CONSISTENT E 30000. NU 0.3,
   MASS_DENSITY 0.000734
```

Example of the commands to specify secondary mass (nodal, element, and secondary):

```
MASS
   NODAL
      2 U V W 20.0  THETAX THETAY 5.0
   ELEMENT MASS FOR TYPE PLANEFRAME
      3 LINEAR  U V W  FRACTIONAL LA 0.25 LB 0.75 WA 3.0 WB 8.0
      1 CONCENTRATED  U V W   L 3.6  M 5.0
      2 CONCENTRATED  THETAZ  L 3.6  M 3.0
   USE LOADING DEAD_LOAD  G 386.4
```

Assembly command:

COMPUTE MASS (FOR) { *STRUCTURE* | *ELEMENT* } *<label>*

Ex: *COMPUTE MASS STRUCTURE TRUSS*

Output command:

OUTPUT MASS (FOR) { *STRUCTURE* | *ELEMENT* } *<label>*

Ex: *OUTPUT MASS ELEMENT WAFER*

A.3.2.2 Specification of Damping - *

Modal damping:

DAMPING MODAL { *RATIOS* / *PERCENTS* } *[<mode list:integer list> <number>]*

Ex: *DAMPING MODAL RATIOS 1 0.01 2 0.015 3-10 0.02*

Rayleigh damping:

DAMPING RAYLEIGH [{ *FREQUENCIES* / *PERIOD* } *<number> <number>* / { *RATIOS* / *PERCENTS* } *<number> <number>*]

Ex: *DAMPING RAYLEIGH FREQ 100.0 2000.0 PERCENT 2.0 5.0*

Output command:

OUTPUT DAMPING { *MATRIX* / *RATIOS* / *PERCENTS* } *((FOR) STRUCTURE <label>) (,)*

((FOR) MODES <integer list>)

Ex: *OUTPUT DAMPING RATIOS STRUCTURE FRAME MODES 1-10*

A.3.2.3 Specification of Frequency Analysis

Definition of the frequency analysis method:

FREQUENCY ANALYSIS (TYPE) { *JACOBI* / *SUBSPACE* / . / . / . }

PROPERTIES <list of properties:label:integer:real>

Ex: *FREQUENCY TYPE SUBSPACE*
PROPERTIES NUM PAIRS 10 ITERATIONS 8 STURM CHECK

Properties for the two analysis methods, JACOBI and SUBSPACE, are listed in Tables A.1 and A.2 respectively.

Computation request:

COMPUTE [*(NATURAL) FREQUENCIES* / *(MODE) SHAPES*] *((FOR) STRUCTURE <label>)*

Ex: *COMPUTE FREQ STRUCTURE FRAME*

Standard output request:

OUTPUT [*(NATURAL) FREQUENCIES* / *(MODE) SHAPES*] *((FOR) STRUCTURE <label>) (,)*

((FOR) MODES <integer list>)

Ex: *OUTPUT SHAPES STRUCTURE FRAME MODES ALL*

Example of mode shape recovery for condensed substructures:

OUTPUT MODE SHAPES STRUCTURE HIGHEST/2/1/2 MODES 1-5

Modal loads output request:

OUTPUT MODAL LOADS ((FOR) STRUCTURE <label>) (,)
(FOR) LOADING <label>

Modal strain output request:

OUTPUT DYNAMIC STRAINS (FOR) STRUCTURE <label> ...

Command	Default	Description
TOLERANCE <number>	1.0E-06	Convergence tolerance
(NUMBER) (OF) SWEEPS <integer>	15	Maximum number of sweeps.
RIGID (BODY) (SHIFT) <number>	.FALSE.	Shift for rigid body modes.

Table A.1 -- Properties for JACOBI Frequency Analysis Method

Command	Default	Description
(NUMBER) (OF) PAIRS <integer>	0	Number of eigenpairs to be computed.
(NUMBER) (OF) ITERATIONS <integer>	0	Maximum number of iterations.
MAXIMUM (FREQUENCY) <number>	-none-	Largest eigenvalue to compute.
TOLERANCE <number>	1.0E-06	Convergence tolerance
SUBSPACE (SIZE) <integer>	function of model bandwidth	Number of iteration vectors to use.
STURM (CHECK)	.FALSE.	Perform Sturm sequence check.
JACOBI (TOLERANCE) <number>	1.0E-12	Convergence tolerance for Jacobi iterations
(NUMBER) (OF) SWEEPS <integer>	15	Maximum number of sweeps for Jacobi iterations.
RIGID (BODY) (SHIFT) <number>	.FALSE.	Shift for ridid body modes.
NOSHIFT	.FALSE.	Surpress positive shifting.
FREEZE (VECTORS)	.FALSE.	Surpress replacement of converged vectors.

Table A.2 -- Properties for SUBSPACE Frequency Analysis Method

A.3.2.4 Specification of User-Supplied Mode Shapes - *

Command sequence:

```
ALTERNATE (MODES)  <name:label> ((TITLE) <string>)

   <specification of DOF order: U V W UX ... >

 [ MODE <mode number:integer>                                  ]
 [ [ <node number:integer> [ <DOF value:number> ]]             ]
```

Ex:
```
ALTERNATE MODES  LAB_TEST
   U  V  W
   MODE 1
     1   0.3  0.0  0.2
     2   0.1  0.0  0.1
     3   0.6  0.0  0.4
   MODE 2
     1   0.0  1.0  0.1
     2   0.0  0.5  0.5
     3   0.0  2.0  0.2
```

A.3.2.5 Specification of Substructure Reduction

Element declaration for intermediate level substructure:

```
ELEMENT 1 TYPE  <structure name:label> CONDENSED (,)

   { RETAIN (NORMAL) (MODES)  <integer list>  }
   { USE ALTERNATE (MODES)  <label>           }
```

Ex:
```
ELEMENT 1 TYPE CHANNEL  CONDENSED  RETAIN  1-10
```

A.3.2.6 Specification of Initial Conditions - *

Command sequence:

```
INITIAL CONDITIONS  <label> ((TITLE) <string>)

 { [ DISPLACEMENTS                                                  ] }
 { [    [<node list:integer list><DOF list:labels> = <number>]      ] }
 { [ VELOCITIES                                                     ] }
 { [    [<node list:integer list><DOF list:labels> = <number>]      ] }
 { USE DISPLACEMENTS ((FOR) STRUCTURE <label>) (,)                  }
 {                   (FOR) LOADING <label>                          }
```

Ex:
```
INITIAL CONDITIONS  PRE_LOAD
   USE DISPLACEMENTS FOR LOADING PULL
```

A.3.2.7 Specification of Dynamic Loading - *

Input of support accelerations as F(x,y,z):

```
LOADING <label> ((TITLE) <string>)
   (NODAL) ACCELERATIONS
      [<node list:integer list> <DOF list:labels> <number>]
```

Ex:
```
LOADING QUAKE
  ACCELERATIONS
    1-3   U   2.0
    1-3   V   1.5
```

Definition of the loading condition:

```
LOADING <label> ((TITLE) <string>)
[ DYNAMIC   ]
[ NONLINEAR ]
```

Definition of G(t) within the dynamic loading condition:

For a harmonic variation of G(t):

```
HARMONIC PERIOD <number> (PHASE (ANGLE) <number>) (,)
      (COMBINE) [ <pattern name:label> (FACTOR) <number> (,) ]
```

For a general variation of G(t):

```
GENERAL (COMBINE) [ <label> [ { TIMES   } <number list> ] ]
                            { FACTORS }
```

For an impulsive variation of G(t):

```
IMPULSIVE (SHAPE) { HALF-SINE     } DURATION <number> (,)
                  { RECTANGULAR   }
                  { POS-TRIANGULAR}
                  { NEW-TRIANGULAR}
   (COMBINE) [ <label> (FACTOR) <number> ]
```

Step size definition within the dynamic loading condition:

```
[ (TIME) STEPS <integer list> ((TITLE) <string>) (,)
          <number list> (SECONDS) ]
```

Selection of the individual steps to save in the data base:

```
SAVE (TIME) STEPS <integer list>
```

Note that the last step computed is always saved, even if not in the integer list or if the command is not given.

Ex: *LOADING VIBRATE*
DYNAMIC
IMPULSIVE HALF_SINE DURATION 0.5 QUAKE 1.0
STEPS 1-100 0.005-0.500 BY 0.005
SAVE STEPS 5-100 BY 5

A.3.2.8 Specification of Transient Analysis - *

Definition of the transient analysis method:

TRANSIENT ANALYSIS (TYPE) { *MODE-SUPERPOSITION* | *NEWMARK* | *CENTRAL-DIFFERENCE* | . . . }

PROPERTIES <list of properties:label:integer:real>

Computation request:

COMPUTE [*DYNAMIC* | *NONLINEAR*] *DISPLACEMENTS ((FOR) STRUCTURE <label>) (,)*

[*LOADING <label> (TIME) STEPS <integer list>* | *INITIAL CONDITIONS <label>* | *INCLUDE MODES <integer list>*]

Output request:

OUTPUT [*DYNAMIC* | *NONLINEAR*] [{ *DISPLACEMENTS* | *VELOCITIES* | *STRAINS* | *STRESSES* } *(<integer list>) (,)*]

((FOR) STRUCTURE <label>) (,)
(FOR) LOADING <label>) (TIME) STEPS <integer list>

A.3.2.9 Specification of Shock Spectrum Analysis - *

Definition of the spectrum:

```
(SHOCK) SPECTRUM  <name:label>  ((TITLE) <string>)
    [ { DISPLACEMENTS  } <number list>
      { VELOCITIES     }
      { ACCELERATIONS  }
    [ { PERIODS        } <number list>
      { FREQUENCIES    }
  DIRECTIONS  (,)
     <node list:integer list>  [ { U } <direction cosine:number> ]
                               [ { V }                           ]
                               [ { W }                           ]
```

```
Ex:  SPECTRUM  SHAKER  "EARTHQUAKE ONE"
       DISPLACEMENTS      0.0     1.0     1.0     0.0
       FREQUENCIES        0.0     5.0   100.0  1500.0
       NODES 1-4
       DIRECTIONS  U 0.5  V 0.6  W 0.624
```

Output request:

```
OUTPUT DYNAMIC [ DISPLACEMENTS          ] (<integer list>)  (,)
               [ VELOCITIES             ]
               [ STRESSES               ]
               [ STRAINS                ]
               [ PARTICIPATION-FACTORS  ]

    ((FOR) STRUCTURE  <label>  (,)

    [ (FOR) (SHOCK) SPECTRUM  <label>                    ]
    [ (FOR) MODES [ <integer list> ]                     ]
    [             [ SRSS           ]                     ]
    [             [ PEAK-SRSS      ]                     ]
```

```
Ex:  OUTPUT DYNAMIC STRSINS 1-100 STRUCTURE FRAME  SPECTRUM SHAKER,
       MODES 1-15, SRSS
```

End of Document